NATIONAL ARCHIVES and RECORDS ADMINISTRATION

INFORMATION TECHNOLOGY

Systems Development Guidelines

Supplement to NARA 805

July 8, 2005
Version: 1.2

Table of Contents

1. Introduction

1.1. Background and Purpose

The NH Architecture Team has developed this *NARA Systems Development Guidelines* document as a comprehensive set of practices used during the implementation and maintenance of Information Technology (IT) systems in the NARA environment.

As the *SDLC Handbook* was developed, several fundamental disciplines were identified for their relevance to the development process. These disciplines include Configuration Management, Software Engineering, Systems Engineering, Risk Management, Security, Records Management, et al. Since elements from each discipline are practiced throughout the development lifecycle they are considered to be of large enough scope that additional detail guidance was deemed appropriate. Guidelines referenced within the *SDLC Handbook* identify where the development process should conform to the practices documented here. A reference in the handbook with the phrase "in accordance with NARA guidelines" indicates where a task should be performed or work product created as outlined in the named section.

The guidelines were developed to provide a framework for project teams during the development lifecycle. As such, they represent the approach desired within NARA; however, the guidelines are generally flexible enough to allow project teams to exercise their own implementation methods. Project teams should bear in mind that a primary purpose of the guidelines is to assist in establishing repeatable practices within NARA. Identifying improvements to the practices is encouraged, but significant deviations should be avoided unless coupled with a justification of the benefit.

1.2. Scope

This document includes guidelines for the practice of various system development disciplines that occur across activities in the NARA Systems Development Lifecycle (i.e., the MED process). They provide greater detail on how certain tasks should be performed within the NARA environment.

The majority of disciplines included in this document have a depth of material that cannot be covered here and already have extensive supporting references that exist on the subjects. The scope of this document is limited to providing the project team with a cursory review and a general process outline for these disciplines with brief explanations and implementation examples. Project team members responsible for items identified herein are encouraged to seek out other source material for detailed discussion of the various practices. A "Suggested Reading List" is included in the *NARA SDLC Handbook* that contains some of this detail information.

1.3. Document Layout

Each section of this document covers either a defined discipline (e.g., Software Engineering, COTS Evaluation, and Systems Engineering diagramming) or a vital practice area (e.g., Security and Records Management). The guidelines typically provide an overview of the discipline or

rationale for supporting a vital practice area, provide a process to implement (if appropriate), identify where to apply certain practices in the SDLC, and provide examples of those practices.

Other disciplines will be added to this document as they are written.

1.4. Related Documentation

The *NARA SDLC Handbook* is a companion document that provides an overview of the activities and tasks necessary in development of an IT system. It describes the Managed Evolutionary Development (MED) process model that is to be used by all IT projects performed by and for NARA.

1.5. Common Terms

Some key terms are used throughout this document for which the reader should have a clear understanding as to how they are applied here.

Product - The resultant system as described by the Product Plan.

Phase - One cycle of a development effort intended to produce a defined outcome. Phase types are defined under the MED systems model as testbeds, prototypes, pilots, initial operations capability, and evolutionary releases.

Project - A temporary effort to create a known set of deliverables

Activity - A discrete portion of a build phase intended to produce certain work products with a milestone review.

Task - A unit of work performed under an Activity. Usually intended to produce a defined outcome or to create input into another task.

Configuration Item (CI) -

An item or set of items managed as a single entity. Changes to a CI are maintained through a single release specification.

Requirements Traceability Matrix (RTM) -

A mechanism used to trace system requirements from their highest level to a built item primarily for the purpose of identifying the impacts of changes or to verify that all requirements are addressed. Can be created manually (e.g., with a spreadsheet) or through Requirements Management tools.

2. Product Evaluation Guidelines

2.1. Purpose

This set of guidelines should serve as the foundation for developing product evaluation plans as well as outline the recommended process for conducting an evaluation. As such, it describes *what* should occur as part of the evaluation process while the evaluation plan will specify *how* it will be done.

During the *Requirements Definition* activity, project teams are to prepare commercial off-the-shelf (COTS) product evaluation plans for each type of product used to implement the defined requirements and system architecture. The evaluation plan itself is partially implemented during the *Requirements Definition* activity as candidate products are identified. The plan and initial candidate products are evaluated for their intended use and likelihood for fulfilling the system needs during the Requirements Review. The remaining implementation of the plan (i.e., the analysis of the products) is performed during the *Preliminary Design* activity.

Though COTS is typically associated with software items, the selection process should be applied to software, hardware, networking / communications, or other components as needed. Additionally, the evaluation process is not restricted to commercial products. It can also be applied to Government off-the-shelf (GOTS) products or reusable components from internal systems.

2.2. Standard Evaluation Process

A. Develop a Product Evaluation Plan

Determine the product type to be evaluated, specify how the product is intended to fulfill system needs, and develop the procedure for evaluating the product (as indicated by the steps below). Product definition during this activity may also trigger revisions to process flows and the system architecture.

B. Candidate product selection

The focus of this activity should be to narrow the field of possible products to a probable few. Identify how candidate products are identified e.g., Internet, trade magazines, expert knowledge, etc. The screening process to be used should be defined within the evaluation plan. A primary goal of this step should be a reasonable assurance that the best possible candidates are identified.

C. Product Evaluation (Functional and Technical)

This section of the plan should include all criteria to be used to evaluate the product in terms of performance, ability to meet requirements, adherence to NARA standards, etc. Specify the basis for what product evaluation criteria are to be used in the analysis. Explain the procedure to be used for selecting the recommended product. The plan may include weighting factors for different criteria with a justification for the weighting.

D. Vendor Evaluation

This section of the plan should include all criteria that will be used to evaluate the product vendor in terms of stability, support, etc. Specify the basis for what vendor evaluation criteria are to be used in the analysis. As with product evaluation, the plan may include weighting factors for different criteria with a justification for the weighting.

E. Write the Product Recommendation

The final activity should be to create a documented assessment of evaluated products as defined by the evaluation plan. The assessment will normally provide pros and cons of each product to determine the worthiness of the recommended product.

2.3. Plan Development and Implementation

A. Develop the Evaluation Plan

Using the recommended activities and tasks outlined in this document, create an evaluation plan that describes how the project team will assess a set of products intended to fulfill a specific need for the target system. This first activity includes steps for defining the product type since this is a precursor to completing the evaluation plan while the other activities and tasks occur after the plan has been developed.

When constructing COTS based systems the approach must be one where tradeoffs are made against available products, the systems architecture, and the functional requirements (or system context). The project team should be prepared to make adjustments to the architecture or business processes while establishing the type of product and developing the evaluation plan. The most expeditious approach is to work each of these tasks in parallel so that adjustments can be made as information is gathered.

The project team should bear in mind that COTS solutions are market driven and do not necessarily lend themselves to custom approaches. When possible, adapt the process to the product rather than vice versa. This will make selection and integration of COTS products more cost effective.

Tasks for creating an evaluation plan:

1. Define a product type
   - 1.1. Select the system architecture component(s) and functional requirements that a recommended product should fulfill
   - 1.2. Identify Enterprise architectural requirements (from the Target Architecture Plan) that products should adhere to. These include:
     - 1.2.1. A well-defined interface (or API) e.g., database systems have established interfaces such as ODBC for passing information and integrating with other components.
     - 1.2.2. A product type that can be acquired from multiple vendors (through competition).
     - 1.2.3. A product type that is reasonably well known in the marketplace and likely to have a cadre of knowledgeable individuals.
   - 1.3. Specify a product type that will likely fulfill the requirements and any features a product should include.
   - 1.4. Make adjustments to the system architecture, business processes, and product type until agreement between them is achieved.
2. Define business process and sample cases to evaluate the products
3. Develop a plan for rating and assessing products. This should include weighting factors for individual selection criteria (outlined in the Product and Vendor evaluation activities).
4. Document the product type definition and the process to be used for producing the final recommendation.

B. Candidate selection process

Recommended evaluation process tasks:

1. Identify boundary constraints and other high level criteria for product selection such as
   1.1. Maximum cost
   1.2. Product complexity
2. Generate a list of available packages that fulfill the product type definition
3. Reduce the list to 3 or 4 products through an informal assessment using the criteria from (1).
4. Document findings and candidate products

Recommended guidelines for candidate selection:

- Select products that use open standards and application programming interfaces (API) rather than proprietary methods
- Opt for mature products:
  - On market for at least 1 year
  - Large installed base
  - Significant market share
  - Influences market trends
  - Availability of consultants / knowledge resources
- Vendor stability
  - Organization
  - Financial
  - Ease of access
  - Reputation
  - History
  - Support infrastructure
  - Engineering approach (standards certification)
  - Maintenance approach
- Are likely to integrate into the NARA architecture with little or no customization
- Able to justify elimination of products that are known marketplace leaders.

C. Functional and Technical Product evaluation

Recommended evaluation process tasks:

1. Perform hands-on evaluation (requires vendor to supply item for evaluation)
2. Incorporate evaluations performed by others (e.g., trade magazines)
3. Evaluate similar organization usage
4. Rate candidate packages according to evaluation plan

Recommended functional criteria:

- Ability to meet functional and performance requirements as tested through business cases
- Adaptability to changing system needs
- Ease of use for users / skill level required
- Ease of learning
- Documentation completeness / Help capabilities
- Error recovery and assistance / self diagnostics

Recommended technical criteria:

- Installed base (Staff expertise available / knowledge base)
- Hardware / OS platforms supported
- Deployment in similar environments
- Customization requirements
- Customization flexibility / impact of upgrades (backwards compatibility)
- Proprietary reliance / Open standards adherence (interoperability, portability, scalability)
- Industry standards adherence
- Target architecture adherence
- Security features
- Product reliability and accuracy
  - Mean time between failure
  - Recovery of faults
  - Assurance of output (including guarantees of results)
- Fit with other products likely to be used in the system / known compatibility problems
- Architecture impact
- Ease of implementation and administration / complexity of the item
- Cost for expected and future usage
- Memory and Storage requirements
- Communication requirements
- Availability of design documentation
- 3rd party tools availability
- Performance
  - Benchmark results
  - Resource usage

- Variable load responsiveness

D. Vendor evaluation

Recommended evaluation process tasks:

1. Prepare requirements criteria for vendors of candidate products
2. Contact vendor references
3. Rate vendors according to evaluation plan

Recommended vendor criteria:

- Help desk and Upgrades technical support
- Upgrade compatibility history
- Ease of upgrades
- Site installation and maintenance support (for customization or ASP services)
- Training support
  - Materials
  - Courses and skill levels covered
  - Customized training
  - Availability
- Financial stability of the company
- Licensing
  - Rights to code / agreements available with contractor for enhancements to product
  - Quantity discounts
  - Transferability of licenses
  - Development / Runtime licensing
  - Type (per seat, CPU, etc.)
  - Standard usage
  - Maintenance
- Support to similar organizations (past performance)
- Ability to provide items within cost estimations / associated risk factors
- Customer satisfaction levels
- Preventive Maintenance

E. Prepare the recommendation

The recommendation process should essentially document the findings of the evaluation and provide the justification for a particular item. Of critical importance in the recommendation is to relate any intangibles that are not reflected in the assessment criteria. The final document should include the following:

- Executive summary briefly describing process, recommended product, and reason for recommendation
- Identify any additional changes to business processes and system architecture caused by recommended product
- Summary analysis of each product with pros and cons
- Product recommendation and justification
- Documented product and vendor evaluations

3. Software Engineering Guidelines

3.1. Introduction

Each project is expected to provide a Software Development Plan (SDP) that defines explicit practices to be followed by the software development team. The engineering practices presented here are intended to provide a framework for creation of the SDP. Several items in this document will specify a minimum standard to be followed for inclusion in the project's SDP. All projects are expected to follow the minimum NARA standards unless justification for deviation is provided.

The SDP should be written to support the Software Management Plan as defined in the Project Plan. The SDP should include the following sections:

1. Development Process Overview

This section provides developers with information on the general development process and how work products are to be produced e.g., through paper documents, CASE tools, etc. This section may also identify the languages and methodologies used during development.

2. Specification Guidelines

This section identifies what requirements and design items are to be documented and the specification style to be used. Products that are used to generate requirements or design models should be given a thorough description of their intended usage.

3. Software Development Environment

Explains the process of how a particular software component will be controlled through CM tools and other methods such as software libraries. It should also explain the environment that developers are to work in with checked out units and the process of check in. Additionally, it should inform developers on how software items are to be turned over for delivery.

4. Coding Standards

Should specify standards for each language that will be used in the system. COTS products should be discussed in use of proprietary vs. open Application Programming Interfaces (API) or other assurances for maintainability of customized features.

5. Quality Assurance Procedures

Describes the reviews (both formal and informal) to be performed. Descriptions of how reviews are to be conducted.

6. Metrics

Identifies the metrics to be collected, when and how they should be collected, and their expected usage.

Software Engineering Guidelines

3.2. Plan Development and Implementation

1. Development process overview

Provide a general description of the software development process in the project including documentation practices, languages, development environments, testing methods, and review procedures. If the project is primarily focused on COTS integration and customization, then the SDP should include details relating to the products being used. This will be especially true in the detailed design, coding standards, testing, and QA processes.

A formal methodology for the software development may be used (such as Structured Analysis and Design, Object Oriented / UML, or Contractor specific) as long as supporting materials are available and the development staff is trained in its use. If the development effort is to be supported by CASE or other tools, briefly describe how and when they are to be used.

When describing the techniques or methodologies to be used, a justification should be provided that explains why those chosen are appropriate for the given application.

2. Specification Guidelines

This section should include instructions for how documents relating to requirements analysis and design should be written for the project. If a formal methodology is being used, then this section should include appropriate links to the methodology's reference material. When CASE or other tools are being used to generate some of the documentation, then the SDP should indicate how that information is to be generated for distribution and configuration control in accordance with the Project Plan.

The SDP should provide guidelines for all of the following documentation needs:

- Software Requirements Specifications

  Include information regarding how the software requirements should be written. An SRS should provide for documenting each of the areas discussed below. Additionally, the SDP should indicate any diagramming / modeling techniques that will be used to support the description of requirements.

  - Problem Analysis and Software Architecture decomposition (the software's functional view)
    - Data flows and transformation processes
    - User interfaces
    - Use-cases

- Logical Data Models and Data Views
  - Address entities, attributes, rules, access, etc. in accordance with documented NARA standards.

- Documenting behavioral specifications in terms of
  - System states
  - Processing actions (stimuli, reaction)
  - Error handling

- Non-behavioral requirements in terms of
  - Reach, Range, Maneuverability (per System Engineering standards)
  - Portability
  - Interoperability
  - Scalability
  - Reliability
  - Security
  - System Resource Efficiency (capacity, performance, etc.)
  - Human engineering (User interfaces / Help / Error messaging)
  - Testability
  - Understandability
  - Modifiability

❖ High-level design

The high-level design documentation should provide details of how the software will fulfill the requirements. The SDP should minimally specify how design should be documented for software components, interfaces (in terms of hardware to software or other integration issues), and the logical and implementation data models. Note that the data modeling must occur in accordance with the NARA standards defined separately.

Recommended items for high-level design:

- Software Components
  - Language or other implementation method such as COTS configurations
  - Installation needs like background tasks, user initiated, distributed operations, etc.
  - Structure (functional routines and subroutines)
  - Security details
  - Function (or method) interfaces (which may constitute a component's API)
  - Module data requirements
  - Process timing (to meet performance requirements)
  - Process flow

- Interfaces
  - User elements and interactions
  - System interface messages, protocols, and triggering events

- Implementation Data Models
  - Address in accordance with documented NARA standards.

❖ Detailed design

Describe how the software development team should document the logical functionality of individual software units. Include any heuristics associated with design of the units that the developers should incorporate such as a complexity limit of 7 - 9 logical paths in a function.

- General error handling specifications
- Algorithm Design
  - Functional complexity guidelines
  - Program logic description
  - Deployment specifications (threading / parallel processing issues, distributed processing issues, platform interfaces)
  - Error capture / recovery
  - Transient data
  - Memory usage
  - Instantiation / destruction of objects
  - Persistent data and file structures
- Physical Data Models
  - Address in accordance with documented NARA standards.

3. Software Development Environment

This section should provide detail instructions for how the software will be worked on by the development team. This section will vary in relation to the need for concurrent development, development team size, and software complexity.

The SDP should address each of the following:

Tools / Development platforms -

Specify the tools and supporting platforms that the developers are expected to use in documentation, construction, and testing of the software. If a separate environment is required from the operational environment then this should be defined including setup procedures for new developers. Testing environments may be transient, but should have a defined procedure for their creation.

Configuration control (version control) procedures -

This should discuss the specifics of how developers acquire code from existing baselines, sharing of software libraries across development teams, and how code is checked back in or added through the configuration control tool. Depending on the complexity of the project,

information regarding concurrent development may need to be included to address merging code revisions. This is particularly true for operations and maintenance environments.

The project's Configuration Management plan may address those items above as well as elements such as build vs. test libraries and lifecycle management to tie delivery of software to known change requests or other documents. The SDP should support elements of the CM plan, but does not need to reiterate them. This section of the SDP should merely present developers with additional details that are not appropriate for inclusion in the CM plan because of the level of detail involved.

Software Deliverables / Development Folders -

Describe all items that should be included as part of a software deliverable package and how they should be delivered. This may include contract specific items like software installable from CD-ROM. Other items may be appropriately maintained in a software library via software development folders (SDF). An SDF may contain such items as meeting notes, correspondence, developer notes, change history, test logs, or any other relevant products of the development process.

4. Programming Coding Standards

When available, coding standards published by recognized organizations such as ANSI or IEEE are encouraged. In some situations, vendor standards may also be appropriate.

Coding standards for each language used on the project should address

- Commenting
- Entry point / Exit point logic
- Function / method size
- Class visibility / object instantiation / destruction guidelines
- Complexity limitations (such as the McCabe complexity guidelines)
- Portability requirements (such as ANSI or POSIX compliant code)

COTS products that require customization via user exits or other means should have guidelines regarding the APIs to be used. For example, if a DBMS allows stored procedures to be written with either proprietary methods or an open standards API, then guidelines should be established for when and why to use to a particular method.

5. Quality Assurance

This section must be written to support the overall project QA plan, but is not intended to reiterate items previously stated. It should be written to inform the developers of what they are expected to do to support the QA efforts. This may include informal or formal peer reviews, test documentation, etc.

6. Metrics

Identify the metrics to be used and how they will be captured during the software development process to support the project management plan. All metrics used within the project should have a corresponding statement of usage in the Project Plan. At a minimum, all projects should collect metrics associated with system size and resource usage (personnel and other) for use in organizational planning.

Example software metrics:

- Product size measured in one or more of the following
  - Number of Requirements
  - Software Size through
    - Function points
    - Lines Of Code (single language comparisons only)
    - Other accepted measure
  - Test Cases / Procedures

- Effort
  - Development hours

- Non-personnel development costs

- Defects
  - Number found
  - Activity point found
  - Rework effort

- Computer resource utilization (estimated vs. actual)

4. Testing Guidelines

4.1. Introduction

A Test Strategy is written to describe what types of testing should be performed on the system and the expected effort necessary to minimize risk of defects. It should include information regarding the development of test plans, creating test data sets, conducting the testing, and resolving test incidents. Part of this description should include specific roles and responsibilities for production of test work products during the project.

Test Plans are written for different levels of testing (i.e., unit, component, system, and acceptance) to describe the tests to be performed. They should be written in accordance with the project's Test Strategy.

The guidelines presented here are intended to provide a framework for creating the Test Strategy. The Test Strategy should indicate the effort required to achieve the needed system reliability. The level of risk associated with system failure or defects must be evaluated to determine the level of effort.

4.2. Test Strategy Development

This section describes the types of testing recommended for possible inclusion in the test plans. Each project should determine which of the testing types is required, how it will be used, and to what extent.

Develop a test strategy that outlines where the testing will occur and identify how these tests will eliminate the risk of failure. The testing techniques included here are not necessarily the only means of testing nor is the project required to conduct all those defined. The test process should be developed to support the type of system being constructed (e.g., batch processing, data presentation, etc.). Certain testing techniques may be appropriately applied at different levels of testing than that defined here. The Test and Evaluation personnel should evaluate and develop the project's process accordingly.

Along with defining the testing expected, the Test Strategy should specify any automation of the process or identify artifacts to be retained. Automation of the process using COTS tools should outline the benefits and usage of the tools used. Artifacts may include any defined metrics, test data, plans, logs, or any other item deemed useful by the development team or product owner.

1. Unit Testing

Description - Unit testing is performed on single functions to ensure that they perform as logically intended. This type of testing is typically conducted on custom developed code and may not be required for COTS implementations unless customization or configurations create the need for unit testing.

Responsibility - Developer usually defines and conducts tests

Test types -

- Structural ("White box") testing: Covers all logical paths defined for the function or method.
- Functional ("Black box") testing: Ensures that the routine produces the correct output for a sampling of inputs.
- Inspection: confirms adherence to programming standards by visual inspection.

Data set construction -

Structural tests should be created by using data to evaluate each logical branch of the code (if - else, do loops, etc.). Use of a matrix to define paths and determine the number of test cases needed is recommended.

Functional tests should be created to match detailed design specifications regarding data transformation, return values, calculations, and results oriented requirements.

Testing considerations -

Unit testing is recommended for all projects as this assures that all code logic is working as expected without "dead code". Unit testing also helps eliminate defects in an earlier and less costly stage.

Testing is often done with drivers and "stubs" for incomplete routines.

Test incident resolution -

Test errors should be documented to assist in tracking areas for process improvement. The developer usually resolves errors in unit and component testing as they are encountered.

2. Component Testing

Description - Usually conducted on a set of software units that together perform a discrete function such as a report, batch program, or user interface. This usually takes place prior to delivery.

Responsibility - Developer conducts testing / Analyst defines tests. Users may assist with testing and reviewing interfaces.

Test types -

- Limit / Domain / Edits testing: checks for invalid conditions, range boundaries, type constraints, and other data inconsistencies.
- Data Integrity / Transaction testing: tests "triggered" data actions, maintaining proper data states during logical transactions, and other events that may cause loss of data integrity.
- Inspection: visual checking of output layouts such as reports or web pages for conformance to specifications and standards.
- Aggregates / Global variables testing: Ensures global and aggregate variables are maintained correctly throughout the application.
- Additional Structural Testing: Ensure procedure calls execute and return correctly
- Additional Functional Testing: Ensure events, calculations, and other data processing performed in accordance with specifications.

Data Set construction -

Develop test cases that will exercise both sides of conditions (i.e., create both "good" and "bad" data). Boundary tests should include a condition on the boundary itself (e.g., if a number is supposed to be <= 10, then have a test at 9, 10, and 11). Use matrices to assist development of test cases and to help avoid redundant tests. Use test cases that cover multiple conditions when possible.

Testing considerations -

Focus on areas of highest risk for failure / impact. Error handling and recovery should be evaluated thoroughly here.

Test incident resolution -

Incidents usually resolved by developer. May require involvement of analyst to assess impact of changes on other components. Use caution in controlling changes to reports, GUIs, and other user interfaces to avoid excessive time on "tweaking".

3. Software Configuration Item or Integration testing

Description - Testing conducted on an integrated set of components and applications that together comprise a configuration item. Focuses on the item's ability to interact and function as a single entity. Usually takes place after components have been delivered and built as a CI. May take place before delivery in instances where a single individual or group is responsible for creation of the CI. This testing may be rolled into system testing unless there are several CI's being developed that would benefit from the separate process.

Responsibility - Developers (before delivery) / Test and Evaluation (after delivery)

Test types -

- Volume / Stress testing: Tests conducted to exercise the CI at peak loads.
- Process Flow and Interfaces: Testing conducted to exercise the communications between internal system components.

Test Data Set construction -

Volume tests often require the use of tools, drivers, or other means to generate large volumes of data. The test cases should be developed to determine if the CI will meet the capacity requirements of the system as defined in the Requirements Specifications (e.g., disk space, query response time)
Interface tests are defined to validate the passing and receipt of information between discrete components or applications within a CI. Tests should be constructed that will validate each element of the interface and any mode or state differences.

Testing considerations -

Volume and stress testing may be difficult to emulate in a development and test environment. Focus on tests that will help identify bottlenecks when possible. Interfaces are a frequent source of problems due to unforeseen issues in the integration of components. Prioritize interface testing towards those most commonly used or present the greatest risk of failure.

Test incident resolution -

The Configuration Item is returned to a development status for the incidents to be resolved. The development team should analyze and correct problems before delivering a new CI. The delivery of the corrected item should be defined as a patch or fix release.

4. System / Release testing

Description - Testing conducted on a system or subsystem to determine if it is operating correctly. Generally performed after Configuration Items are delivered for a system release. May be dependent upon external interfaces to conduct testing.

Responsibility - Test and Evaluation personnel

Test types -

- Requirements Testing: Testing conducted to ensure that all defined functional requirements have been met.
- Intersystem testing: Testing conducted with external systems to ensure transfer and receipt of data according external interface agreements.
- Recovery testing: Testing to ensure that disaster recovery plans can be implemented effectively.
- Security testing: Tests conducted to ensure that the desired level of security has been achieved.
- Process Flow and Interfaces: Testing conducted to exercise the communications between internal system components.

Data Set construction -

Business process cases (or use-cases) should be defined in order to test the requirements. Matrices should be used to help develop cases that can cover more than one requirement, avoid redundancy, and ensure that all requirements have been covered.

Testing considerations -

Critical / mandatory requirements should be given priority. Testing must be coordinated with external systems for getting participation in the test. Interfaces between internal components ought to be tested if not done so previously.

Test incident resolution -

The development team should correct incidents through patch or fix releases. Significant discrepancies may require negotiation for postponement of a correction to a later build phase or a change to requirements. Changes to problem components should return to a development activity to ensure that they are tracked and managed appropriately.

5. Acceptance Testing

Description - Formal testing conducted with users to ensure that the system meets defined acceptance criteria (which may differ from specified requirements). The testing is often very similar to the system test, but with a focus on the user's ability to interact with the system. This testing will also evaluate any users guide or operators manual for completeness.

Responsibility - Users / Operators / Test & Evaluation

Test types -

- Business processes / Use cases: Tests using operational scenarios. Will also help assess sufficiency of user documentation.
- Manual support testing: Tests constructed to assess the sufficiency of operator documentation and the ability to support typical production operations.

Data Set construction -

May be able to use test cases from system testing. Test cases should be constructed for both successful and error conditions to determine user and operator ability to recover from problems.

Testing considerations -

Test cases should be identified that are most likely to address the Product Owner's concerns. This requires that the development team work closely with the PO and other users to establish acceptance criteria and specific test cases. The intent in acceptance testing should not be to demonstrate that all requirements have been met; system test results will be the basis for establishing that defined requirements have been addressed and it is not typically necessary to do so again here.

Test incident resolution -

Incidents occurring during acceptance testing should be evaluated for their impact to acceptance and production. In some cases, assessment of the problem may determine that the system may be implemented in its current form and the issue is addressed during the next build phase (or system release). This should be negotiated with the Product Owner after determining the expected time to correct any issues.

Issues that affect the developed system (as opposed to user and operator materials) should be returned to an appropriate development activity to ensure that changes are tracked and managed appropriately. Corrections to the established release should be applied as a patch or fix release version.

6. Regression testing

Description - Testing conducted on changed items using previously executed tests. Usually selected in relation to the scope of the change using a mix of other strategies. Regression testing should always be conducted when a portion of the system has changed to make assurances that correctly operating items have not been impacted.

Metrics

The test strategy should identify how the test process will be monitored and measured to determine the effectiveness of testing.

Before establishing which metrics are to be collected, the Test Strategy must indicate the objectives to be gained and how they plan to evaluate whether those objectives have been met. Some common metrics gathered during testing include

- Cost of test in terms of work hours, computer resources, tools, and any other applicable resources.
- Defects uncovered during testing
- Defects uncovered after testing
- Defects uncovered by activity
- Time to correct
- etc.

5. Systems Engineering Diagramming Methodology

5.1. Introduction

This section defines the NH-approved approach for diagramming reach, range, and maneuverability characteristics for client-server systems planned to be developed and integrated at NARA. The intent of the Appendix is to present a comprehensive approach to communicate client-server architectures in an unambiguous manner to achieve interoperability across NARA IT systems and networks. This is NOT INTENDED to be a substitute for actually performing the systems engineering and the systems design nor for providing training to NARA personnel in the discipline of systems engineering. A relatively high-level of sophistication and experience on the part of the reader in the area of IT systems engineering is assumed. It is intended to provide a diagrammatic method for experienced systems engineers to communicate important technical design parameters for IT systems to achieve interoperability.

The systems engineering diagramming approach presented in this section was originally developed at the Bell Labs, and has been used at Exxon, IT&T, and NASA Goddard Space Flight Center. Within this document, reach, range, and maneuverability characteristics are defined as follows:

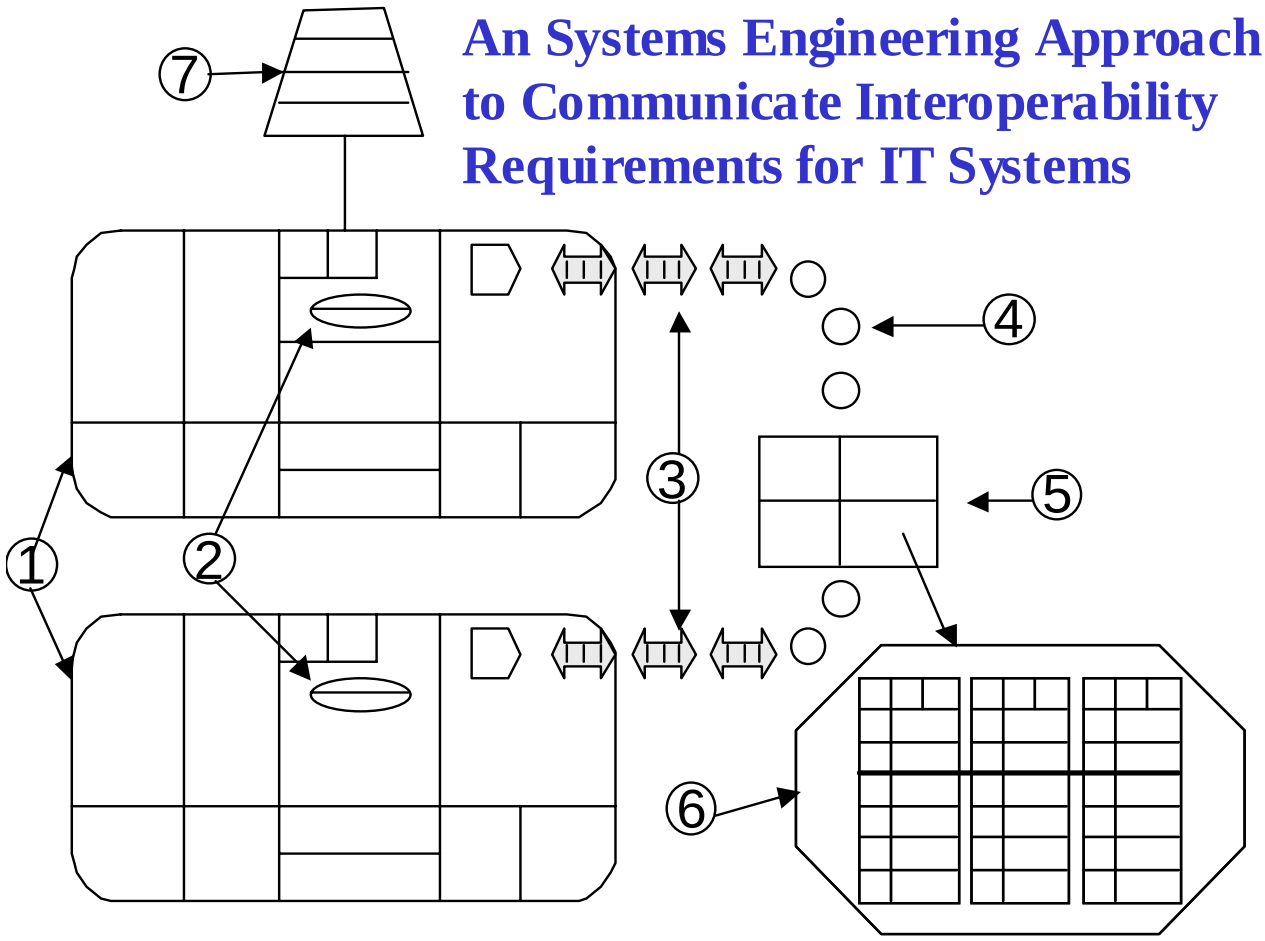
- **Reach:** Reach defines who can access information, from where, with what security, with what interface, and in what mode (e.g., wireless, wired, etc.)
- **Range:** Range defines what forms and structures of information (e.g., simple files, text databases, multimedia databases, messages, distributed databases, etc.) can be accessed
- **Maneuverability:** Maneuverability defines factors that help protect investments in current IT solutions while providing less risky methods of incorporating emerging technologies. One important aspect of maneuverability is designing systems to be scaleable and reusable.

The NARA systems engineering diagramming methodology is a comprehensive and proven graphical method of presenting the architecture of an IT system. In implementing the systems engineering diagramming methodology, NH will provide training and assistance to NARA IT project managers and project teams. At the current time, the major limitation of the systems engineering diagramming methodology is that it is a manual process. That is, there are currently no tools to specify the major architectural components within an IT system (and across interfacing systems), to manage the components as part of an integrated product database, and to automatically produce the systems engineering diagram. NH advises that the Architectural Team (AT) will initially use Visio templates to develop the systems engineering diagrams. Additionally, the AT will also provide training and assistance. As the NARA IT Architecture is implemented, NH anticipates that integrated and automated tools will be identified, evaluated, and standardized across NARA.

At the current time, project managers and project teams only need to be familiar with the concepts in this section at the level necessary to work with the AT and to task expert developers and integrators with providing the technical details needed to "flesh out" the diagrams. Initially, the AT will coordinate with the project team to understand the high-level interface and interoperability requirements for proposed new products and to capture and render this information in the diagrams produced near the start of a product development effort (such as the Product Plan and the Concept of Operations document). As the project progresses, the systems engineering diagrams will be refined towards the goals of reach, range and maneuverability for the proposed product. NH anticipates that the technical details to achieve interoperability and such systems goals will be largely provided by system developers and integrators, who are recognized "industry experts". The AT will work with the project team and contractors to develop and maintain the systems engineering diagrams over the lifecycle of the product. As such, the systems engineering diagrams will become a formal part of the product baseline for configuration management purposes.

In practice at NARA, the systems engineering diagramming methodology will be initially applied at three (3) levels (or tiers) as part of the IT product and planning process. A 4^{th} tier to accommodate common and reusable approaches for operations, administration, and maintenance (OA&M) of IT systems is reserved for future development. NH anticipates that as the NARA Target IT Architecture is gradually developed and implemented, NH will specify the use of common and reusable OA&M architectural components as part of the future NARA Technical Reference Model (TRM).

Figure 5-1 depicts a schematic of an architectural drawing at Level 3. Level 3 is the most detailed level and is appropriate for detailed systems design as part of a Critical Design Review (CDR) Level 1 will be highly simplified and only show the high-level connections of the proposed product with other systems with which it must interface. Level 1 diagrams will be most appropriate for early product planning documents such as the Product Plan and Concept of Operations. Level 2 diagrams will offer more detail than Level 1 diagrams. Level 2 diagrams will be most appropriate for Preliminary Design under the Waterfall model. Figure 5-1 is the basis for development of the architectural drawings at lower levels.



An Architecture drawing shows:

1. Configured platforms
2. Program layers (software, program name, and layers)
3. Service path
4. Interoperability
5. Attributes (middleware and APIs)
6. Internet working option
7. Important resources related to project

Figure 5-1. Drawing Overview.

An architecture drawing shows the configured platforms, service paths, and networking that comprise the distributed computing environment.

As depicted in Figure 5-1, the major components of the diagramming methodology include the following:

- **Configured Platforms** interoperate with each other as information technology devices (ITDs). An ITD is any kind of IT system, information appliance, computer, network station, or dumb terminal that may participate in a client/server architecture. A configured platform is an ITD processor with an associated operating system, on which one or more program layers operate. At a high-level, appropriate to the NARA Product Plan and the Concept of Operations document, an IT system (to be designed, developed, implemented, and integrated) is also considered a configured platform. The reader should note, however, that while an IT system can be considered a configured platform within this context, the notion of a configured platform is a much broader and more descriptive concept. Indeed, as a product design and development effort progresses and as a system gets more well defined, configured platforms will likely need to be broken out into finer detail. This breakout will particularly be the case for complex distributed systems, where servers are perhaps geographically distributed, and/or specific functions are allocated to particular processors to achieve required performance levels. For any given product development, the Level 2 and Level 3 diagrams will need to address this breakout and allocation of functionality across configured platforms.

- **Program Layers** are executable programs that are written in one or more program languages. If correct design practices are followed, a program layer does only one program function, namely presentation, processing, or data management. Within the NARA IT Architecture, a specific program layer may take the form of a COTS product to be embedded within an IT system. Appropriate to the NARA Product Plan and the Concept of Operations document, the Level 1 diagram should have a minimum of one (1) program layer to describe at high, global level what the product is intended to accomplish. Level 2 and Level 3 diagrams will normally breakout the program layers into more discrete program layers to reflect the allocation of functions to specific program layer configuration items (CIs). Examples of program layers on a given configured platform might include: a data input routine, a statistical package, a visualization and rendering tool, a search tool, an office automation product, etc. In Level 2 and Level 3 diagrams, the program layers will effectively identify and define the software configuration items. As noted, one of the functions on a given configured platform might be allocated to a COTS product, assuming that the COTS product has a proper application programming interface (API).

- Program layers interoperate with each other through **Service Paths.** A service path is a finite set of service calls that can communicate both physically (via a network) and semantically with a service path on another configured platform. Defining service paths is the most complex aspects of the systems engineering diagramming methodology process. The difficulty arises because it is at this “bit and byte” level of communications that most systems that are not interoperable fail. In the Level 1 and Level 2 diagrams, the service paths will be represented by simple arrow annotated to indicate what information needs to be passed between systems (Level 1) and between program layers (Level 2). In the Level 3 diagrams, the service paths between configured platforms and program layers will be fully resolved.

- An **Interoperability Path** is illustrated by a filled circle. These filled circles are merely provided to show the connectivity and are visual artifacts to show the connections.

- An interoperability path has certain attributes, such as the **Middleware Framework** used, the slowest speed of the path, whether it supports transactions, a pointer to a definition the network used to communicate between services paths, etc. Defining the Middleware Framework within a complex IT system is also tantamount to defining the Application Programming Interfaces (APIs) between the program layers. Defining the APIs is one of the most important aspects of IT systems engineering to achieve interoperability. The detailed middleware and APIs need to be defined in the Level 3 diagrams.

- An **Internet Working Diagram** shows the internetworking options chosen for each network traversed to connect the service path. Within the scope of this *Developing IT Solutions* document, an Internet Working Diagram broadly includes intranets (such as NARANET), extranets, virtual private networks (VPNs), and the Internet (itself). Within this Appendix, the term "internet" refers to a network of networks. The detailed Internet Working Diagram will be a part of the Level 3 systems engineering diagram.

- **Important Resources**, such as databases, enterprise printers, CD-ROM production facilities are identified and related to the program layers and the configurations that manage them.

5.2. Tailoring of the Diagramming Methodology within NARA

Within NARA, the diagramming methodology presented in Figure 5-1 will initially be applied at three (3) levels, with a fourth level to accommodate operations, administration, and maintenance (OA&M) of IT systems reserved for future use. The essence of the approach to developing a systems engineering diagram to convey reach, range, and maneuverability characteristics is to break the proposed new system down into its component pieces as illustrated in Figure 5-1 and to "fill in the blanks" in the diagram template. By breaking the new system down into components, the systems engineer can determine how the components interoperate both within the new system as well as with components in other interfacing systems. As the product development effort progresses, the basic requirement will be for the project team to become "more specific" in defining the major characteristics of a system to achieve interoperability and to meet mission requirements for the proposed product.

In reference to Figure 5-1, the following defines the tailoring to be accomplished for each of the three (3) systems engineering diagramming levels:

- **Level 1 Diagram.** Figure 5-2 depicts the Level 1 template for the systems engineering diagramming methodology. A Level 1 diagram shall be developed and provided for the Product Plan and the Concept of Operations.

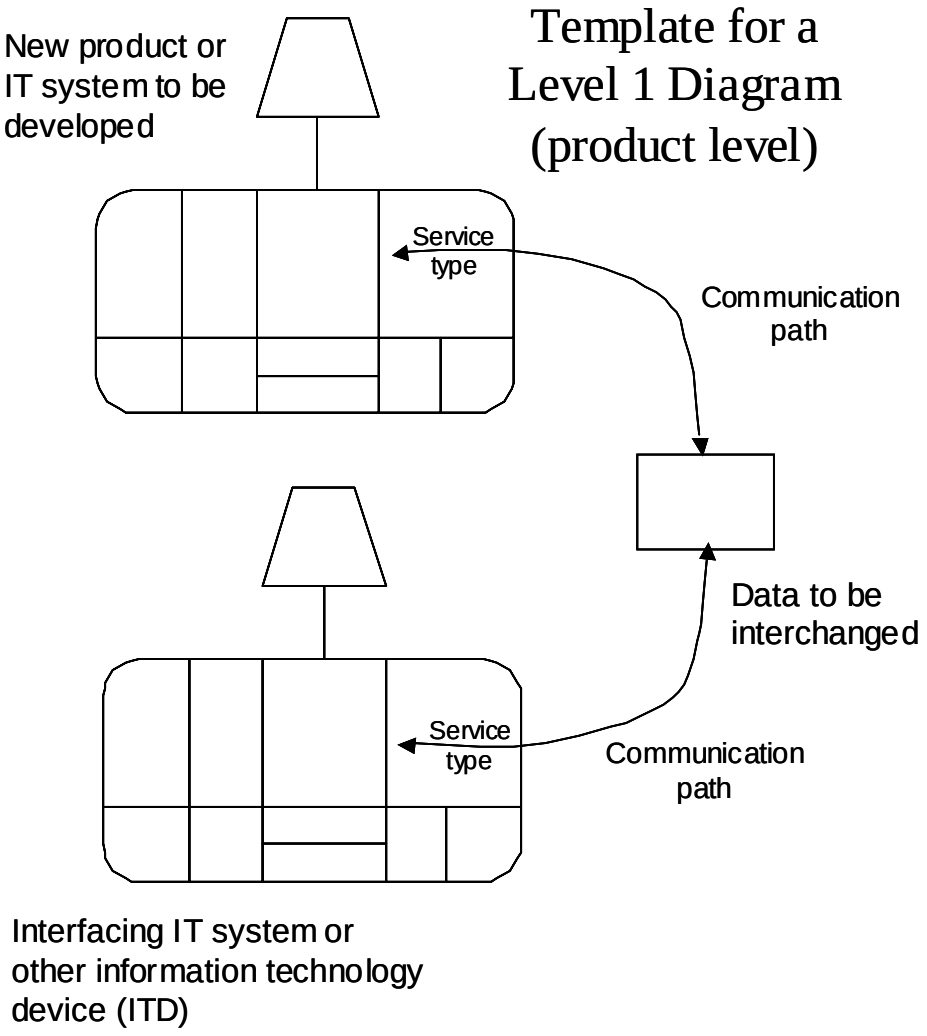


Figure 5-2. Template for a Level 1 Diagram (Product Level Description)

The Level 1 diagram relates the new product to be developed with other systems or products with which it must interface. The Level 1 diagram is intended to depict (at a high-level) the following information:

- Name of the product or system
- Major product function
- Important resources related to the product such as databases
- Simple descriptive information (e.g., physical location, device type, operating system (if known at the time of developing the diagram), user data, etc.)
- Communication paths

- Data to be interchanged with interfacing systems
- For each interfacing system:
    - Name of the interfacing system
    - Major function(s) to be interfaced with the product
    - Important resources to be used by the product on any interfacing system
    - High-level service to access the interfacing system
    - Simple descriptive information for the interfacing system

Such information is routinely captured for the Product Plan submittals. The Level 1 diagramming approach merely provides a convenient method to visualize the required connectivity and high-level functionality of the proposed product.

- **Level 2 Diagram.** Figure 5-3 depicts the Level 2 template for the systems engineering diagramming methodology.

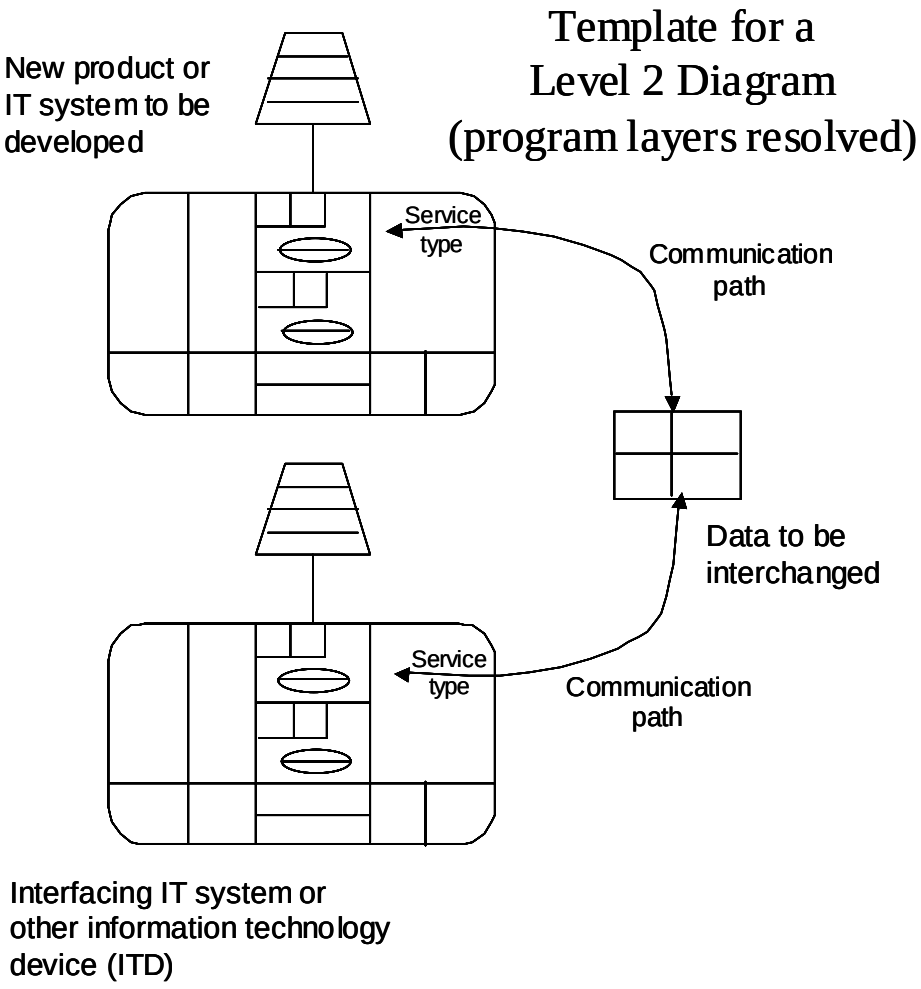


Figure 5-3. Template for a Level 2 Diagram with Program Layers Resolved

The Level 2 diagram shall depict the major program layers, the layers in any important product resources, and the major components in the data to be interchanged. A Level 2 diagram shall be developed and provided as part of the preliminary design for the new product and for any Engineering Change Proposals (ECPs). The Level 2 diagram shall be addressed at the Preliminary Design Review (PDR) for the new product or for any ECPs. The Level 2 diagram shall be derived from, and be consistent with, the Level 1 diagram. The Level 2 diagram will provide additional resolution on the following:

- Identification of all the program layers in the product and any necessary program layers in any interfacing systems (including layer type, layer instance#, application, and source language)
- Identification of all major information technology devices (ITDs) comprising the new product or system. An example of a type of ITD might be a platform such as a server, a client, a workstation, a mainframe, etc.
- Further breakdown on important resources within the new product and any important resources in any interfacing systems (including resource type, resource name, layer instance number, and commit-group)
- Further breakdown on the data to be interchanged between all program layers on the new product with any program layers on any interfacing systems (including identification of services to be invoked, middleware, required transaction processing support, networking speed range, and any internet-working requirements)

- **Level 3 Diagram.** Figure 5-4 depicts the Level 3 template for the systems engineering diagramming methodology. In the Level 3 diagram, the service paths shall be fully resolved between all program layers in the product with any program layers in any interfacing systems. A Level 3 diagram shall be developed and provided as part of the detailed design for the new product and for any Engineering Change Proposals (ECPs). For configuration management and control purposes, a Level 3 diagram shall also be prepared for the final as-built product in the Version Description document. The Level 3 diagram shall be addressed at the Critical Design Review (CDR) for the new product and for any ECPs. The Level 3 diagram shall be addressed at the Production Readiness Review (PRR) for the Version Description document.

  The Level 3 diagram shall be derived from, and consistent with, the Level 2 diagram and provide the necessary full resolution of the service path(s) connecting program layers. The Level 3 diagram will provide additional resolution on the following:

  - Identification of numbers, type, sequencing, and “stringing together” of service structures within service paths between all program layers on the new product with any program layers on any interfacing systems. Information on each of the service structures shall include the: service type, the service interface, the service interface level, and the service provider. NARA project managers are advised to consult with the Architecture Team to identify any

constraints on the allowable sequencing of service structures as part of service paths. There is significant potential for the service paths to be very complex.

- Identification of any service path prefixes. Service paths that involve interoperability between program layers require a service prefix before the header services. The service prefix contains information on the interoperability domain and the interoperability role for the service path.
- Further resolution of the middleware description to include the preparation of the internet-working diagram.

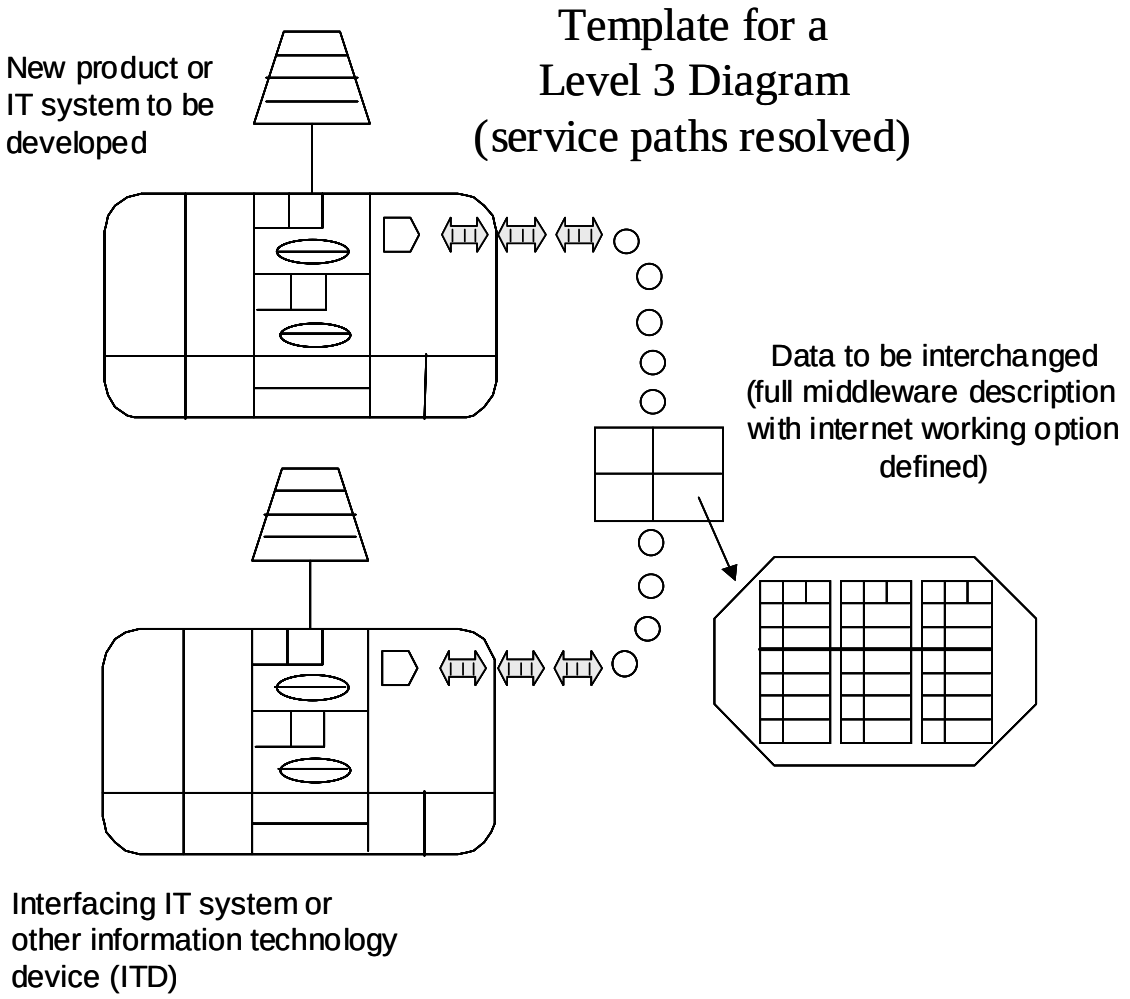


Figure 5-4. Template for a Level 3 Diagram with Service Paths Fully Resolved

5.2.1. Information to be Entered into the Systems Engineering Diagramming Templates

The following identifies the information to be entered into the various "parts" of the systems engineering diagram for the three (3) levels defined above. In preparing the Level 1, 2, and 3 diagrams, the various parts to be specified include the following:

- Specification of Configured Platforms (or Configuration Items)
- Specification of Program Layers as Part of the Configuration Items
- Specification of Important External Resources Related to the Configuration Item
- Specification of Services and Service Paths Linking Configured Platforms (and/or Configuration Items)

1. Specification of Configured Platforms (or Configuration Items). Figure 5-5 depicts the high-level template for configured platforms (or configuration items) to be defined as part of the systems engineering diagramming methodology for Levels 1 - 3.

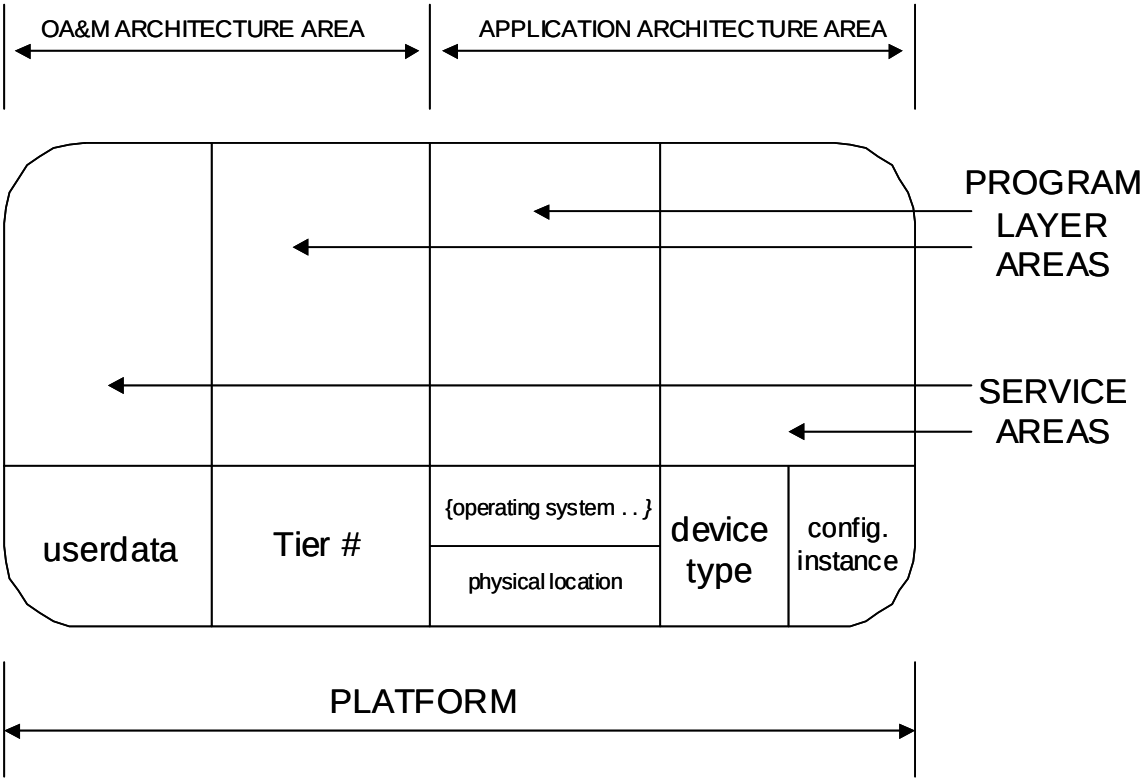


Figure 5-5. Template for Configured Platforms (or Configuration Items)

Figure 5-5 generally applies to new products (or to new systems) in the Level 1 diagram, and to specific platforms (or information technology devices) comprising the new product in the more detailed diagrams for Levels 2 and 3. In other words, configuration items may be the entire product, a system, a subsystem, a configured platform (such as a workstations or server), etc. For example, a proposed new product (or a new IT system) may be comprised of separate component information technology devices (ITDs). Examples of ITDs that might comprise the new system (or the new product) and be in geographically diverse locations might include: multiple distributed servers, secure enclaves, clients, database machines, printers, etc.

The Level 1 diagram only needs to provide the configured platform template shown in Figure 5-5 once for the high level product proposed to be developed. The Level 1 diagram should also provide a similar template for every other system or component for which the new product will interface.

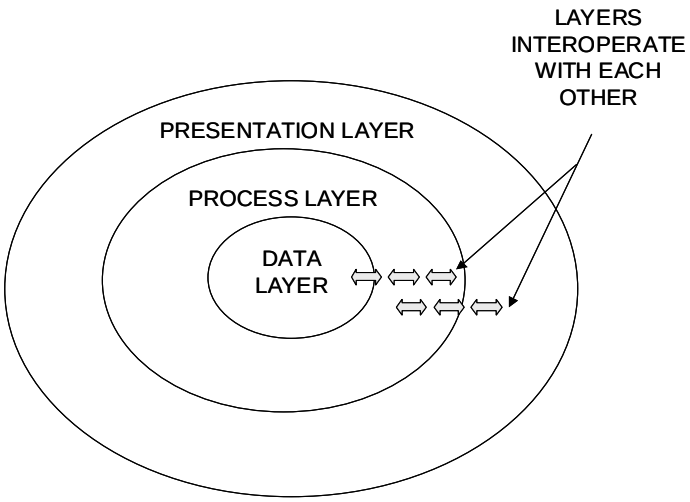
For Level 2 and Level 3 diagrams, a separate configured platform template (as depicted in Figure 5-5) must be provided for every major ITD or component comprising the proposed new product.

As shown in Figure 5-5, the configured platform template consists of an "OA&M architecture area" and an "applications architecture area". Each of these architecture areas is further subdivided into "program layer areas" and "service areas". Additional space is provided in the template for entry of user data, Tier#, operating system, physical location, device type, and configuration instance. These attributes are defined in more detail below. At the current stage of development of the NARA Systems Engineering Diagramming Methodology, the OA&M architecture area in Figure 5-5 can be left blank. As noted above, this area is a placeholder for future development as part of a future Level 4 diagram.

Before providing detailed guidance on filling out the template depicted in Figure 5-5, it is important to understand the nature of program layers that might comprise a new system. Any IT application should be understood as consisting of three (3) basic program layers that can interface with each other. These layers are the essence of a 3-tiered IT architecture for an IT system. These layers are:

- **Presentation Layer (PN):** the layer that handles application logic with regard to collecting, preparing, and presenting data.
- **Processing Layer (PR – a.k.a the Function Layer):** the layer concerned with application business logic.
- **Data Layer (DT):** the layer concerned with data management (including addition, deletion, selection, and modification of data records).

The following figure illustrates the concept of layering of the presentation layer (PN), the processing layer (PR), and the data layer (DT): This partitioning into three major layers is the basic structure of all modern IT systems.



Within the scope of this *Developing IT Solutions* document, the NH and the Architecture Team strongly recommend that new IT products and systems be designed and developed as 3-tiered architectures with the PN, PR, and DT layers separate and distinct from each other. Such separation should be reflected in the Level 2 and Level 3 diagrams as the project progresses to preliminary and critical design review. It is not necessary to depict such separation of presentation, processing, and data layers in the initial Level 1 diagram as part of the Product Plan and ConOps Document.

Separation of the program layers is currently the best way to design software to promote interoperability, portability, re-use, and re-configuration. As a practical matter, especially with legacy systems and a number of COTS products, it is also important for the Project Manager and Project Team to realize that the PN, PR, and DT layers are frequently intertwined and combined. Combinations of the program layers are possible including PNPR, PRDT, and PNPRDT. For example, PNPR indicates that the presentation and processing logic within a given application is intertwined and inseparable within a program. This is a common, but bad practice, for many COTS products.

In filling out the template in Figure 5-5 for configured platforms (and configuration items), the following minimum information should be provided in the space located in the template. In all instances, the information to be provided should be as explicit and definitive as possible.

- **OA&M Architecture Area** – leave blank
- **Application Architecture Area – Program Layer Areas.**
    - In the Level 1 diagram, indicate the name of the application or the product to be developed. A Level 1 diagram does not need to explicitly identify the specific program layers in the applications area in the template depicted in Figure 5-5. For systems that will interface to the proposed new product, also provide the name in a separate template.
    - In the Level 2 and Level 3 diagrams, this area should contain a more formal description of the program layers and their separation into PN, PR, and DT

components. This requirement to provide additional resolution in the program layers is addressed in more detail below.

- **Application Architecture Area – Service Areas.**
    - In the Level 1 diagram, provide only high-level statements of the type of service to be developed to interface the proposed product with other external systems. Examples of high-level statements include (but are not limited to): manual interface, automated interface, user interface, file transfer services, messaging services, remote procedure call, security services, etc. Additional examples of service types (with abbreviations are provided in Table 5-1). Within the Level 1 diagram, the Project Manager and Project Team should be as explicit as possible in defining the proposed service type for any interfaces for the proposed new product with other external IT systems. In reference to Figure 5-2, the Level 1 diagram should also identify the major communications path (examples: NARANET, Internet, Departmental LAN, dial-up, etc.) and provide a bulletized list of the major data to be interchanged with any interfacing systems.
    - In the Level 2 diagram, with the program layers further resolved in reference to Figure 5-3, provide similar high-level statements of the types of services to be developed to interface each of the program layers to specific program layers in all of the external IT systems, with which they must interface.
    - In reference to Figure 5-4, for the Level 3 diagram, the service paths and communications paths must be fully resolved. This requirement to provide additional resolution is discussed in more detail below.
- **User Data** – In the diagrams for Levels 1 – 3, this area in the template depicted in Figure 5-5 is reserved to be populated with any data the Project Manager and Project Team user would like to supply about the configuration of the platform or ITD (or its building blocks). Such information should be provided in a bulletized list format in the appropriate area. Possibilities might include: required MIPS ratings, required storage capacities, required security attributes, or any other information deemed important to define the high-level configuration and its requirements. For the Level 1 diagram, this information refers to the proposed product. For Levels 2 and 3, this information refers to specific configuration items.
- **Tier#** - In the diagrams for Levels 1 – 3, this area in the template depicted in Figure 5-5 is intended to convey information about accessibility for the configuration item. Once again, a Level 1 diagram refers to the accessibility of the entire product. For Level 2 and Level 3 diagrams, the accessibility refers to the particular configured platform and/or ITD comprising the product. In filling out this item in the template, the Project Manager and Project Team should select from the following list:
    - Tier 1: mobile (nomadic) information technology devices
    - Tier 2: desktop information technology devices
    - Tier 3: Departmental, workgroup, or NARA building level processing and data servers
    - Tier 4: Widely-distributed NARA processing and data servers
    - Tier 5: Widely-distributed processing and data servers across the NARA enterprise involving external business partners

- **Operating System.**
  - In the Level 1 diagram, the operating system for the product is considered optional. If the operating system is known, specified, or explicitly required, then it should be entered in the appropriate place on the template. Otherwise, for Level 1 diagrams, this area may be left blank.
  - For Level 2 and Level 3 diagrams, enter the operating system into the appropriate location on the template depicted in Figure 5-5.
- **Physical Location**. For Level 1 – 3 diagrams, enter the physical location (or geography) where this configuration item is located. The specification of the physical location should be as explicit as possible to define the location of the configuration item.
  - For widely distributed product development efforts documented in Level 1 diagrams, the physical location can be specified in collective terms such as "all NARA facilities", other federal agency facilities, specific departmental workspaces, Archives 2, enterprise-wide, etc.
  - For Level 2 and Level 3 diagrams, where the configuration item has been more defined, the physical location should be more concrete and more explicit – down to a room within a specific building, if possible.
- **Device Type.** For Level 1 – 3 diagrams, enter the make and model of the hardware to run the configuration item. The entry should be as explicit, as possible. It is acceptable to provide an entry in collective terms, such as personal computers, engineering workstations, parallel processors, etc. For Level 3 diagrams, the specific make and model should be specified.
- **Configuration Instances #.** – For Level 1 – 3 diagrams, the Project Manager and Project Team shall indicate the number of instances of this configuration item in the proposed architecture.
  - For Level 1 diagrams, where a single monolithic system is to be built, the value to be entered in the template is "1". In Level 1 diagrams, for products that will be replicated in their entirety and distributed across the enterprise, enter the approximate number of times the product is to be replicated.
  - For Level 2 – 3 diagrams, enter the number of configuration instances of the specific item. Please note that if a particular set of applications, that are otherwise common, are run on different devices and/or are run from different physical locations, these items should be handled distinctly different configuration items and not be counted in the total.

2. Specification of Program Layers. Figure 5-6 depicts the high-level template for specification of program layers as part of the configured platforms (or configuration items). The program layer notation identifies the type of layer, the instance number of the layer on the particular configuration, the name of the application, and the source languages that compose the application.

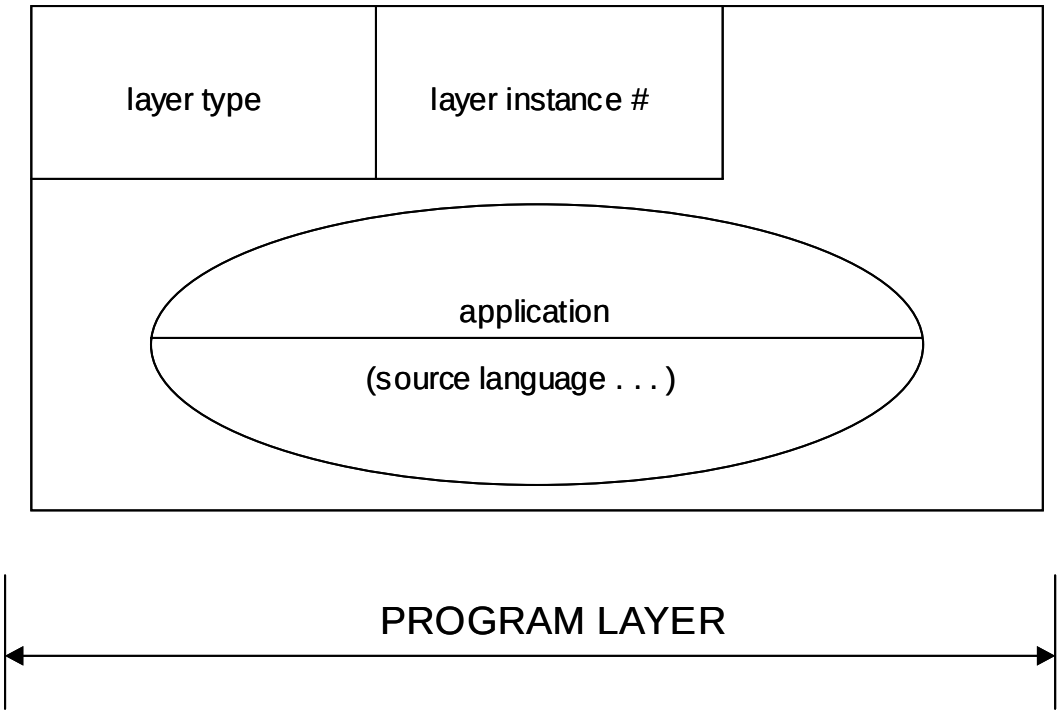


Figure 5-6. Template for Program Layers

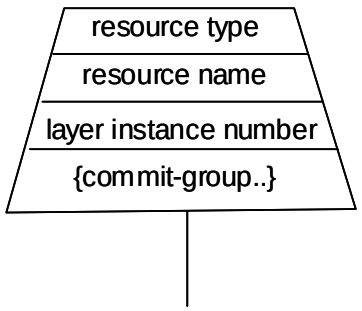
In reference to Figure 5-6, program layers are executable programs that are written in one or more computer languages. If correct design practices are followed, a program layer does only one program function: namely, presentation (PN), processing (PR), or data management (DT). While it is possible to combine and bundle program layers, the Project Manager and Project Team are advised to consult with the NH Architecture Team to determine the pros and cons; as well as to explore the alternatives.

In reference to Figures 5-2 and 5-3, program layers must be specified and resolved for the Level 2 and Level 3 diagrams. Specifying program layers for Level 1 diagrams is considered optional – just specify the major application for the product in the Level 1 diagram. For Level 2 and Level 3 diagrams, the major application for the product needs to be functionally decomposed into various program layers and the program layers need to be allocated to specific configured platforms (or configuration items). Level 2 and Level 3 diagrams must identify all of the program layers in the product and any necessary program layers in any interfacing systems. Service paths (discussed in more detail below) are the major mechanism (or "glue") through which program layers on different configured platforms interoperate.

As depicted in Figure 5-5, a configured platform (or a configuration item) is an entity on which 1 to n program layers are executing. Further, a given program layer may exist 1 to n times for 1 to n different applications. Figure 5-6 (above) illustrates the notion for a single program layer on a given configured platform (or a given configuration item). The major systems engineering and design parameters to be specified in the Level 2 and Level 3 diagrams for each program layer include the following:

- **Layer Type:** identifies the type of layer (i.e., PN, PR, DT, PNPR, or PRDT). Note that a bundled layer (i.e., PNPR) on a configured platform is not the same thing as two distinct PN and PR layers on the same platform. A PNPR layer is one intertwined glob of logic, which the PN and PR layers are individual, distinct programs that can be manipulated individually. The Architecture Team advises that separating PN, PR, and DT layers is an important architectural concept for new product developments because it vastly influences portability, reconfigurability, and interoperability.
- **Layer Instance #:** an integer that uniquely identifies this program layer on this configuration. It has no other significance other than as a numeric identifier.
- **Application:** the name of the application running in this program layer. For product development efforts, where COTS products may be employed, the application name might be the COTS product name, or perhaps a middleware component name.
- **Source Language:** a list of source languages from which this program layer is built (i.e., C, C++, COBOL, BASIC, etc.) For program layers, where the application is a COTS product or a component of a COTS product, the Source Language should specify the language (or languages) for an application programming interfaces (APIs) for the COTS product/component. At this stage in the development of this *Developing IT Solutions* document, NH advises that a list of allowable source languages for systems development may be specified in the future as part of NARA's Technical Reference Model.

3. Specification of Important External Resources Related to the Configuration Item. Figure 5-1 shows how important external resources related to the configuration item can be incorporated into the diagramming methodology as a networking resource. The most prevalent type of resource is a relational database; though scanners, large (color) printers, CD-ROM production facilities, and other enterprise resources can also be readily incorporated in the diagram. By diagramming database servers (as an example), the project team can visually display where such a resource is located, what configured platform hosts it, and what program layer controls the resource. The following figure is the template for an external resource.



In diagramming external resources, the following are the major attributes to be entered into the template:

- **Resource Type.** Indicate the type of resource (e.g., database, color printer, CD-ROM burner, etc.)
- **Resource Name.** Indicate the business name of the resource (e.g., employee database, metadata database, archival catalog, etc.)
- **Layer Instance Number.** Indicate the number of the program layer on the owning configured platform (or configuration item) that controls the resource.
- **{commit group…}**. These are the names of two-phase commit groups that this resource belongs to. The NARA project team should consult with the NH Architecture Team for additional details. This field in the template is mostly intended to depict necessary database functionality associated with complex transaction processing and e-commerce applications, where data integrity is a critical design objective.

In diagramming external resources, the following tailoring guidance is provided:

- Level 1 diagrams only need to identify any important resource types and their names (if known) at the product or system level.
- Level 2 diagrams need to identify the all of the external resources and their names; and associate the resources with specific program layers.
- Level 3 diagrams need to address all of the data fields including the {commit group…}.

4. Specification of Services and Service Paths Linking Configured Platforms (and/or Configuration Items). Within the NARA systems engineering diagramming methodology, service paths are the primary vehicle or mechanism for program layers to communicate with each other across configured platforms. The concept of services and service paths is the most complex aspect of the systems engineering diagramming methodology. The specification of services and service paths is critical to achieve interoperability among systems. This section presents a simplified overview of the diagramming approach for specifying services and service paths. The NARA Project Manager and Project Team are encouraged to consult with the NH Architecture Team to more fully understand and explore the complete set of options for defining services and service paths, especially for complex IT systems.

Within NARA, this document strongly supports the fundamental software development notion, especially prevalent in networking computing, that applications programs should confine their "interest" to the business application or required function at hand. When applications programs have need of major software functions (especially common and reusable functions), they should invoke them through service requests. The service (e.g., transaction processing, data access, messaging, electronic data interchange (EDI), remote procedure call (RPC), security services) should be invoked through a standardized and well-designed application programming interface (API), or by knowledge of the protocol format.

Within the NARA systems engineering diagramming methodology, the concept of a service interface hides the inherent complexity of delivering the service to the requesting program layer, makes the program layer dependent only on defining the interface, and most importantly with network computing, hides all of the interoperability issues. When a whole set of services is so

bundled to deliver rich interoperability services to program layers, the overall service is called *"middleware"*. See Figure 5-1. It follows that for middleware to be successful, it must bridge at least two (2) services paths for program layers between configured platforms to make them be interoperable.

Templates. Figure 5-7 depicts the template for fully defining and resolving services and service paths. As shown in Figure 5-4, this template is appropriate for the Level 3 diagram. Figure 5-8 depicts a simplified version off this template for identifying the high-level services and service paths appropriate to Level 1 and Level 2 diagrams.

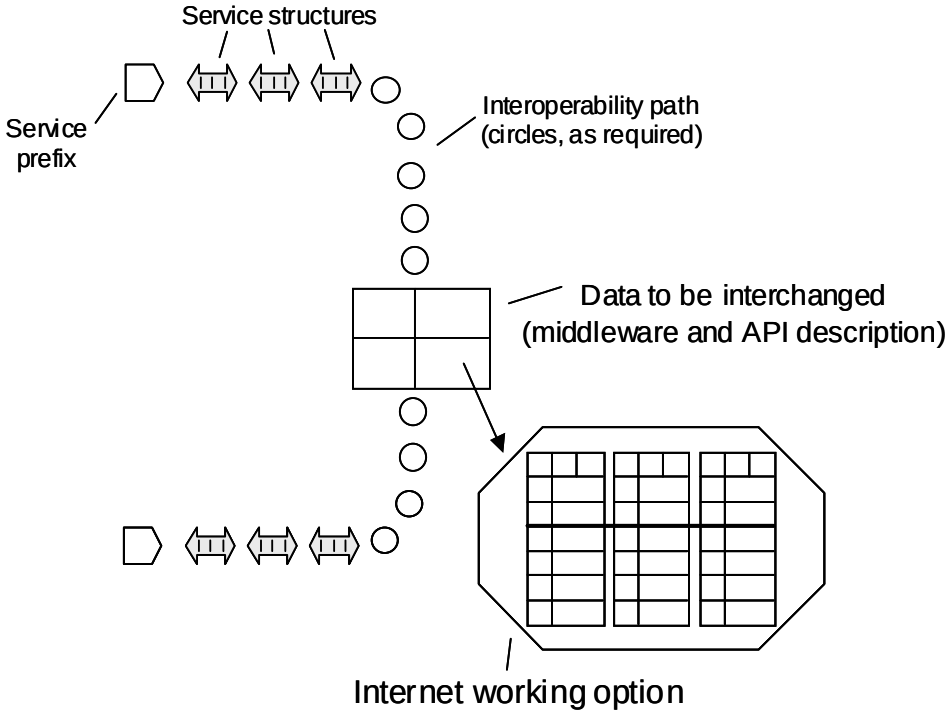


Figure 5-7. Template for Specification of Services and Service Paths (for Level 3 Diagrams)

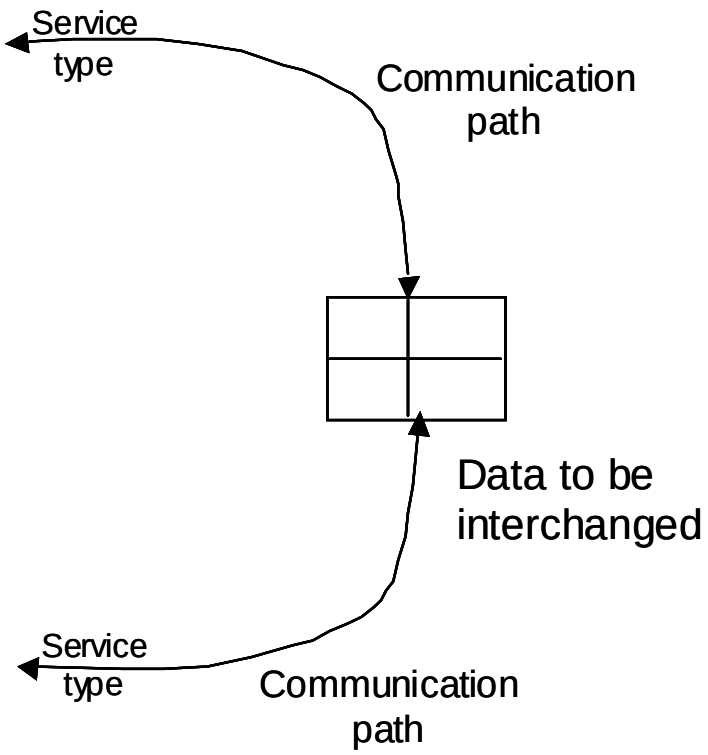


Figure 5-8. Simplified Template for Specification of Services and Service Paths (for Level 1 and Level 2 Diagrams)

Concept of Services and Service Structures. Within the NARA systems engineering diagramming methodology, a program layer within a configured platform can avail itself of major software functionality by invoking services. In reference to Figure 5-7, these services requests are defined in the diagramming methodology via the "Service Structures" template. The detailed template for defining service requests via the service structures is depicted in Figure 5-9.

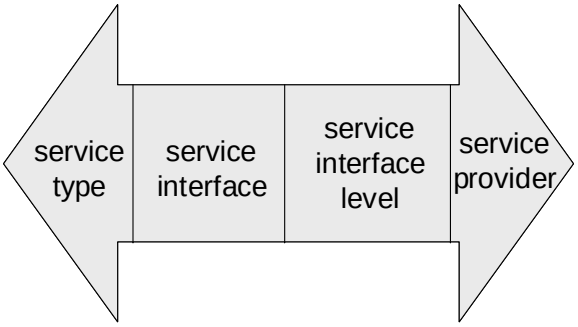


Figure 5-9. Template for Defining Service Requests via Service Structures

As shown in Figure 5-9, a service request consists of four (4) components. A modified service structure, as illustrated in Figure 5-10 is also possible to support the case where a service

provider provides two (2) service interfaces for a service type. Such a situation is common when a service supports both a proprietary legacy API and a new standard API.

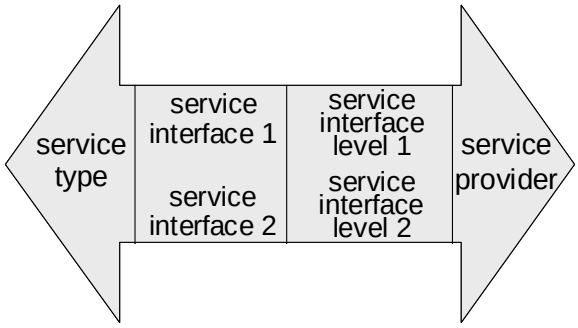


Figure 5-10. Modified Service Structure

The four (4) major components of the service structure include the following:

- **Service Type.** The service type identifies a unique class of service to be invoked. Table 5-1 provides a list of the candidate types of services to be considered (with their abbreviations). If none of the entries in Table 5-1, the Project Manager should consult with the NH Architecture Team to propose a new candidate service.
    - For the simplified Level 1 and 2 diagrams, the major, high-level service type should be indicated as an annotation to the arrow connecting the program layers. See Figure 5-8 and Figures 5-2 and 5-3.
    - For Level 3 diagrams, the abbreviation for the service type shall be entered into the appropriate space in the service structure template as shown in Figure 5-9.

- **Service Interface.** The service interface identifies the interface that is being invoked. For example, for the transaction management (TM) service type, possible service interfaces could include:
    - Tuxedo API
    - ATMI
    - Cooperation API
    - IDMS Embedded Language
    - IMS Call Interface CICS Embedded Language

  Another example of a service interface would be the UNIX File Copy command for the File Transfer Service (FT) service type. As the current time, a detailed list of service interfaces have not been fully defined for all of the candidate service types identified in Table 5-1. The NARA Project Manager and Project Team should consult with the NH Architecture Team to identify and specify appropriate service interfaces. This requirement applies to the development of the Level 3 diagram.

Table 5-1. Identification of Candidate Service Types

Service Types	Service
TM	Transaction management services
CI	Communication services
RPC	Remote procedure call services
DA	Data access services
OS	Operating system services
TE	Terminal emulation services
OP	Output services
UI	User interface
TR	Transport services
EDI	Electronic Data Interchange services
MSG	Messaging services
FT	File transfer services
DTM	Distributed transaction management services
SEC	Security services

- **Service Interface Level.** Within the service structure template depicted above in Figures 5-9 and 5-10 above, a service interface may be defined at one of the three levels of transparency in the following order of increasing transparency:

    - L1: The networking protocol and location of the invoked program layer must be known.
    - L2: Only the location of the invoked program layer must be known.
    - L3: Neither the network protocol, nor location of the invoked program need be known. This service interface level is ideal for later re-configuration of the architecture. One example of such a service interface level would be a look-up of protocol and/or program location via a database or other directory services.

- **Service Provider.** The service provider identifies the software that provides the service interface. For example, the Cooperation API is provided by the Cooperation products from AT&T, and the ATMI interface is provided by Tuxedo. File transfer service providers are typically the vendor of the operating system and the operating system name (e.g., Microsoft Windows 2000).

Service Paths. As illustrated in Figure 5-7, service structures may be strung together to form a service path. A given service may invoke from 0 to *n* other services in a recursive manner. This means that a service, in turn, may require utilization of another downstream services. A special case exists for the service invoked directly by the program layer. This initial service (expressed in the diagramming methodology as a service structure) is called the *header service*. As illustrated in Figure 5-11, the concatenating or stringing of services (viz., the service structures)

can get to be very complex with various Boolean "and" and "or" possibilities. The NARA Project Manager and Project Team are strongly advised to consult with the NH Architecture Team to receive additional training on service paths and to discuss the possibilities. As a practical matter in the systems engineering of complex systems, the inordinate stringing together of distributed services can also have a significant impact on overall systems performance.

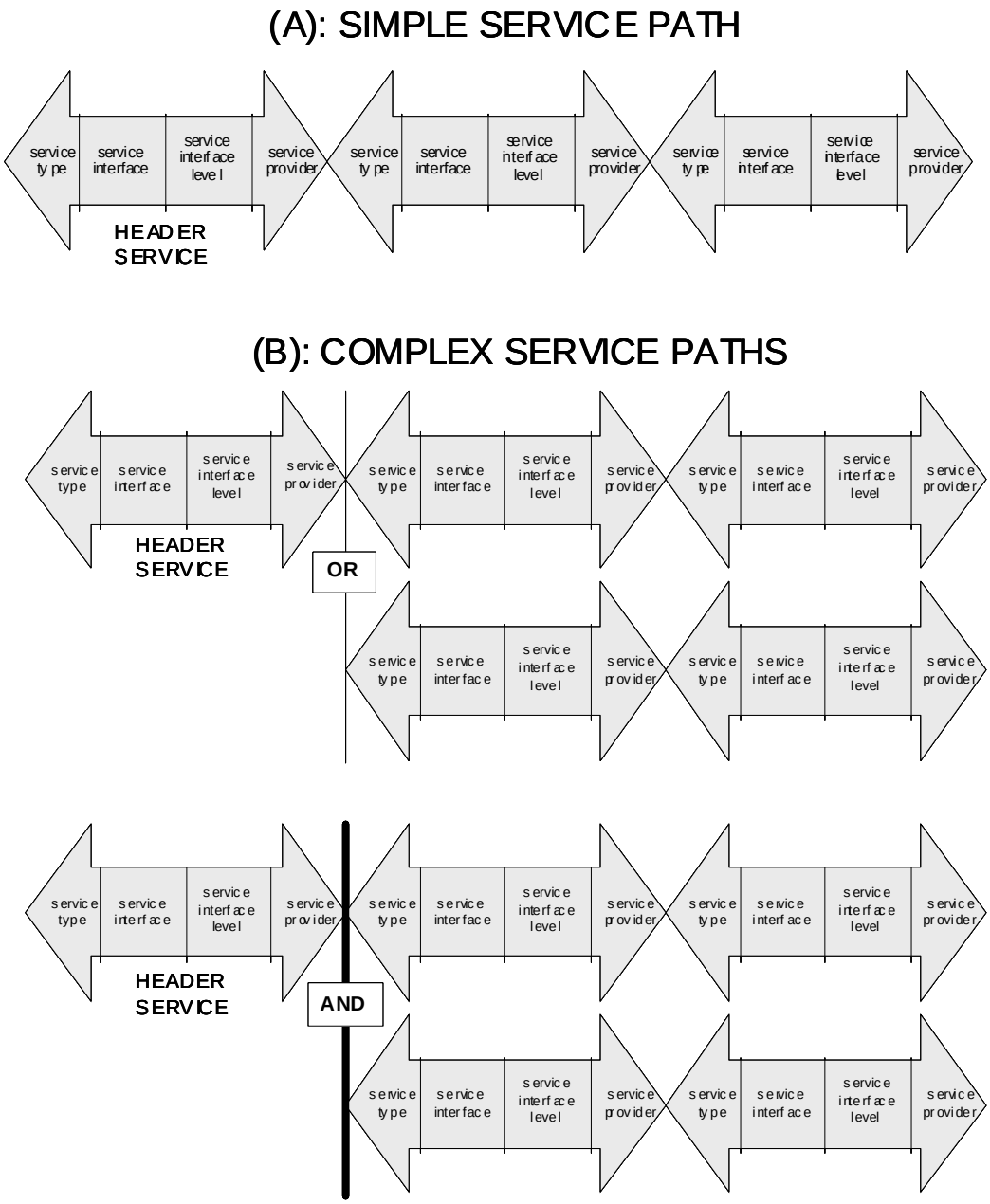


Figure 5-11. Examples of Service Paths

Service Path Interoperability and Service Prefixes. Within the NARA systems engineering diagramming methodology, a given service path may or may not involve interoperability. A

service path that does not require interoperability with other program layers (on the same or other configured platform) is called an *insular service path*. Such service paths should be drawn for documenting the overall configuration of the system and/or for detailed systems design purposes.

A service path that requires interoperability between program layers on the same or a different configured platform is called an *interoperability service path*. In systems engineering, this situation is the norm. As illustrated in Figure 5-7, an interoperability service path requires a service prefix before the header service. The template for the service prefix and its relationship to the header service is depicted in Figure 5-12. As illustrated in Figures 5-7 and 5-8, the service prefix only needs to be depicted for the Level 3 diagram.

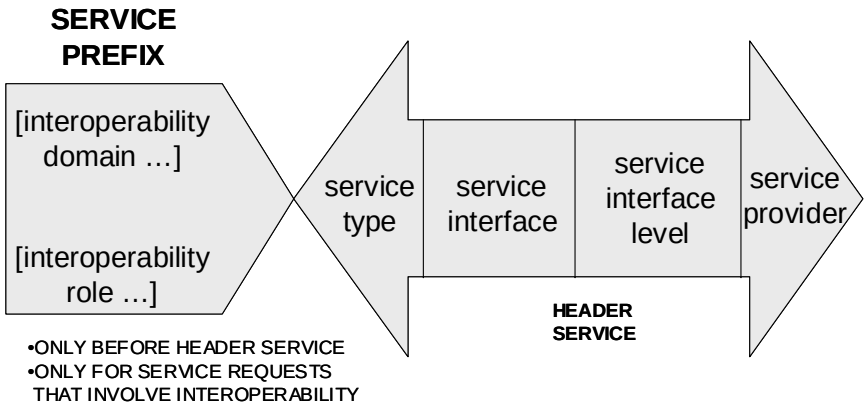


Figure 5-12. Template for the Service Prefix

In reference to Figure 5-12, and for Level 3 diagrams (only), the major attributes of the service prefix template include the following:

- **Interoperability Domain.** Program layers may interoperate with each other only within defined execution domains. The interoperability domain name should be entered into the template in the appropriate position. A domain defines an interoperable execution space. The most common interoperability domain is the concept of the Windows NT "workgroup". Security and firewall circumstances considered, an interoperability domain (such as a specific Windows workgroup) can, in principle, be operative across the Internet, across NARANET (e.g., NARA's Intranet), or across an Extranet. A program layer may interoperate within multiple domains concurrently. When two service paths interoperate, they must share a common domain, or interoperability cannot take place.
- **Interoperability Role.** A program layer invokes a service with an intended interoperability role. Enter the appropriate code form the following list into the service prefix template. Within the NARA systems engineering diagramming methodology, the permitted roles include the following:
    - CL: client

- SR: server
- CLSR: client or server
- PE: peer
- SD: sender (in a file transfer sense)
- RR: receiver (in a file transfer sense)
- SUB: subscriber
- PUB: publisher

The NARA Project Manager and Project Team should note that, for two service paths to interoperate, they must share a pair of permitted interoperability roles. For example, a client can interoperate with a server; but not directly with another client. Peers (by definition) can interoperate directly with each other. In invoking a service path, a program layer may have more than one choice of roles consistent with the capabilities defined by the header service. In preparing this portion of the Level 3 diagram, the NARA Project Manager and Project Team should consult with the NH Architecture Team to identify and discuss allowable combinations of interoperability roles.

Interoperability - Relationship to Middleware and the Internetworking Drawing. As a matter of systems engineering to achieve interoperability, it is important to realize that program layers interoperate with each other only through service paths in certain permissible combinations, and only with permissible networking restrictions. These combinations and networking restrictions tend to be very complex to understand and are perhaps the largest single source of concern and frustration in designing and developing IT systems to achieve interoperability across the enterprise. In general, across the NARA enterprise, transactions and information flow in a future distributed IT systems environment will typically traverse multiple networks as illustrated in Figure 5-13.

The NARA Project Manager and Project Team are strongly advised to consult with the NH Architecture Team prior to undertaking the detailed design of the product/system. Specifically, the discussion needs to address the role of *middleware* as well as the *Internetworking options* that are available to meet the high-level product/system requirements within a distributed networking environment.

As illustrated in Figures 5-1 and 5-7, the detailed role of middleware and the detailed networking options are documented in the Middleware template and the Internetworking Options template, respectively. Together, these templates address the “connectivity” and networking implicit in connecting the program layers as shown in Figure 5-13. Figure 5-14 depicts the middleware template. Figure 5-15 depicts the network template for user data or information traversing a single network. Figure 5-16 depicts the template for user data or information traversing multiple networks (e.g., the internetworking diagram). An internetworking diagram is a sequence of network diagrams that shows the ordered paths of networks traversed between communicating program layers.

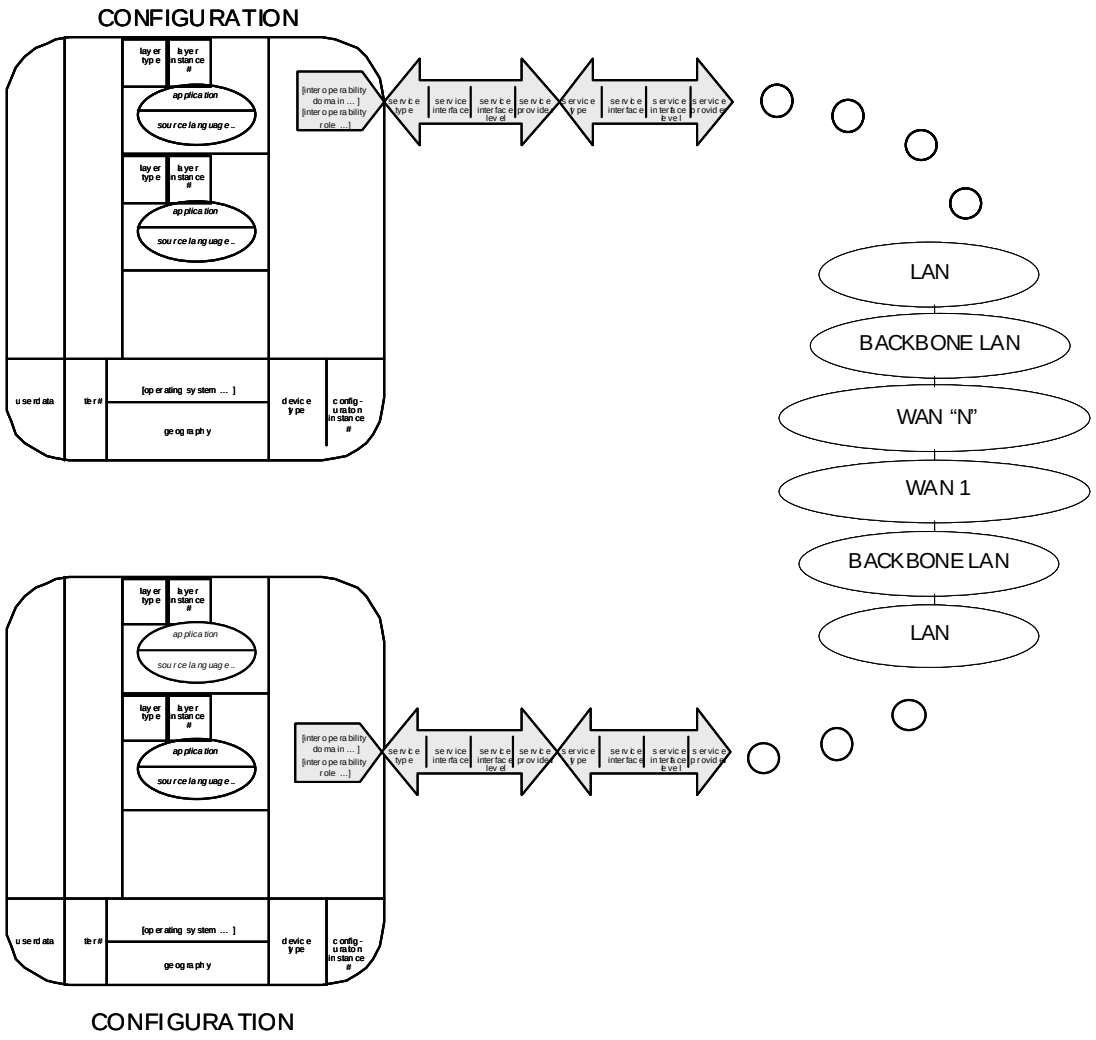


Figure 5-13. Illustration of Information Flow for Program Layers across Multiple Networks.

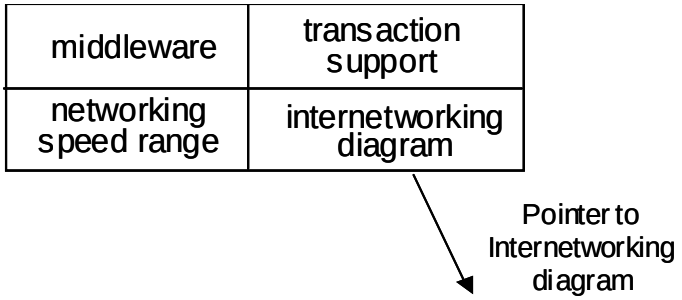


Figure 5-14. Template for the Middleware Box

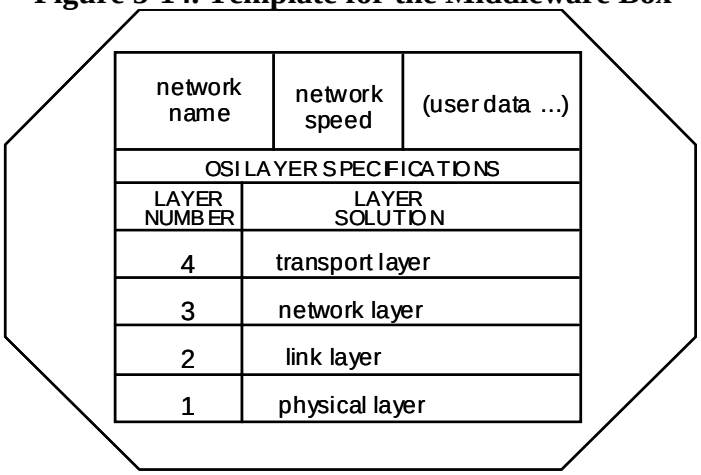


Figure 5-15. Detailed Template for User Data or Information Flow Traversing a Single Network

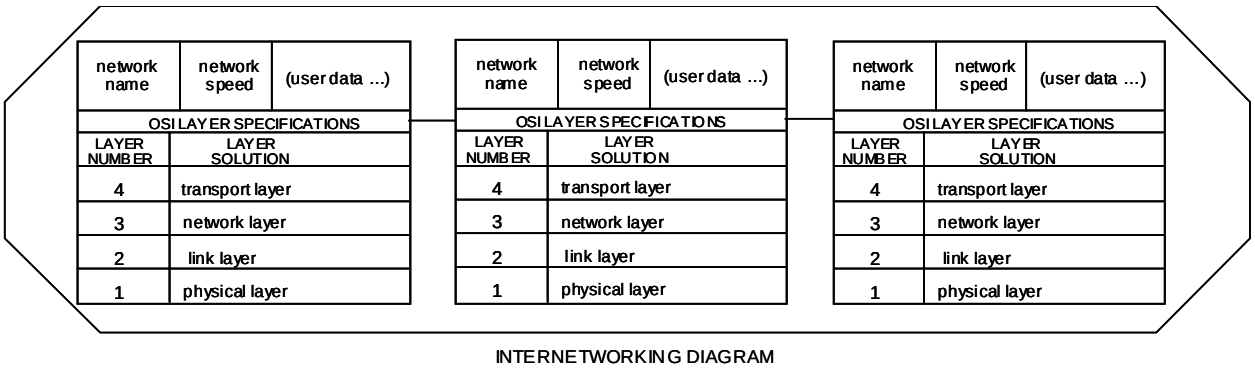


Figure 5-16. Detailed Template for User Data or Information Flow Traversing Multiple Concatenated Networks

Discussion of the Middleware Template. Figure 5-14 depicts the template for the middleware box. The middleware box furthers describes the conditions of interoperability among program layers on the configured platforms. In reference to Figure 5-14, the major attributes of the middleware box include the following:

- **Middleware Field.** This field defines the middleware framework that oversees the connection between the program layers. Table 5-2 identifies the major classes of middleware and some specific frameworks for each class. The middleware framework attests to the validity of the two service paths to work with each other. For Level 2 diagrams, enter the name of the class of the middleware into the

appropriate field into the template. For Level 3 diagrams enter the name of the specific middleware framework to be employed.

Table 5-2. Identification of Middleware Components

Note: In the table below, the classes of middleware and the exemplary frameworks are not necessarily complete. NARA Project Managers and project teams are encouraged to consult with the NH Architecture Team for additional information and guidance

Classes of Middleware	Examples of Specific Frameworks
Distributed File System	RFS, NFS
Messaging	VIM, MAPI, POP, cc:Mail, AOCE, CMC, X.400 (MHS), SMTP
Distributed Function (Processing)	RPC, DCE (RPC), APPC, Sockets
Distributed Database	DRDA, ODBC, RDA, EDA/SQL, ODAPI, SQL-Access
Distributed Transactions	X/Open-ATMI, Tuxedo, Transarc, Top-End
Object Request Brokers	CORBA
Mainframe Message Switching	MSC. ISSC
Distributed Presentation	X Windows, MS Windows (WinAPI)
Directory Services	X.500, LDAP
Terminal Emulation	3270 Emulation, Async Terminal Emulation
File Transfer	FTAM, FTP, UUE, MIME, transportable media
Network Management	CMIS, SNMP, Netview, DME
OSI Transport Level Interface	IPX/SPX, NetBios, UNIX TLA, TCP/IP
Telephony	AT&T/Novel Telephony Interface
Electronic Data Interchange (EDI)	X.12, EDIFACT
Intra-processor Interoperability	DDE, OLE, UNIX IPC (messages, semaphores, shared memory, named pipes)
Security	SSL, PPTP (for VPNs)
Public Key Infrastructure (PKI)	X.509

- **Transaction Support.** The transaction support field is intended to indicate whether this interoperability event supports formal transaction processing, or not. The concept of transaction processing can be very complex, yet needs to be addressed at the time of preliminary design and be refined during the critical systems design process. This field in the template is an indicator “YES” or “NO” for whether formal transaction processing is supported or not. Within this field, such transaction processing support shall be indicated by one of the following notations: “T=YES”, “T=NO”, or “TBD”. The value “TBD” is reserved for Level 2 diagrams, where the project team may still be uncertain as to whether formal transaction processing is required or not for a given service path.

The concept of transaction support applies primarily, though not exclusively, to database updates ("Begin Work", "Commit Work", and "Abort" type commands), and the ability of middleware to support formal transaction processing is frequently critical to the design of on-line transaction processing systems, operation support systems, and business process automation systems. Basically, a *transaction* is a unit of work that must meet all of the following conditions for the transaction support indicator in the middleware box to be set to the value of "T=YES":

- **Atomicity.** Either the entire unit of work is completed, or none of it is completed. If a partial amount of the work is completed, the atomicity test fails, and the transaction support field should be set to "T=NO".
- **Consistency.** The object of the unit of work moves from one consistent state to another consistent state. If it remains in the original state, then the transaction support field should be set to "T=NO".
- **Insulation.** The unit of work executes as though it were the only unit of work. If otherwise, then the transaction support field should be set to "T=NO".
- **Durability.** Once the unit of work is completed, the consequences of the unit of work are, in fact, the "new reality". In other words, a "firm commitment" has been made. If this firm commitment has not been made, then the transaction support field should be set to "T=NO".

- **Networking Speed Range.** This field in the middleware box indicates the minimum required lowest and recommended speed for the set of networks the transactions will traverse in this interoperability event. Please see Figure 5-13 for an illustration of information flow between program layers that traverse multiple networks. Unlike in traditional host-centered computing, where transactions generally traverse one wide-area network (WAN), in network computing and in client-server environments, it is common for a transaction to traverse a set of networks (e.g., departmental LAN, enterprise backbone LAN and WAN, Internet, ISDN dial-up network, etc.). Each of these networks can impose speed restrictions that can have a significant impact on overall throughput and performance.

  Each network may have different transmission speeds and gateway (e.g., switching) speeds. Such network speed, often referred to as bandwidth, is an important IT systems design factor. The networking speed field in the middleware box is populated with the minimum required lowest and recommended speed for the multiple set of networks that are traverse as part of this service path. The next field, the internetworking diagram, gives the name of a specific diagram that shows the internet working that will be employed for the product/system and the associated speed of each individual network. The network speed range is customarily expressed in Kbps or Mbps. It is also acceptable to express network speeds in industry terms such as T-1, *n*T-1 (e.g., multiple T-1s), DS-3, OC-3, OC-12, etc.

- **Internetworking Diagram.** The Internetworking diagram field in the middleware box is the name of, or a pointer to, a separate diagram that shows the sequences of networks to be traversed as part of this service path. The internetworking diagram

template is depicted in Figure 5-15. A sample extension of the internetworking diagram template for information to traverse three (3) concatenated networks is depicted in Figure 5-16.

The internetworking diagram template (Figure 5-15) is discussed in more detail below. The template contains a *user data* field. As illustrated in Figure 5-7, a complete internetworking diagram only needs to be prepared for a Level 3 diagram. As illustrated in Figure 5-8 and discussed in more detail below, the internetworking field in the middleware box is simplified and tailored for the Level 1 and Level 2 diagrams to only identify the specific data, information, and/or transaction to be passed via the service path. In addition, the names of the proposed networks (e.g., communications paths) to be traversed for the service path will be annotated on the arrows connecting the program layers.

Discussion of the Internetworking Diagram Template. Figure 5-15 depicts the network diagram template. In discussing this template, the reader is assumed to have a basic understanding of systems engineering for networks and the 7-layer Open Systems Interconnect (OSI) model. The OSI reference model is a layered set of generic functions that every network must fulfill. The lower levels, levels 1 –4, define the network interface. The higher levels, levels 5 – 7, largely define the details of the application interface. In applying the NARA systems engineering diagramming methodology and for interoperability purposes, it is most important to understand the sequence of networks traversed at the network interface level of the OSI model. Specifically, the NH Architecture Team is most interested in the interface of the higher-level application program layers to OSI Level 3, which is the network layer. The interface of the applications to the network layer is via Layer 4, the Transport Layer.

Figure 5-16 shows a representative application of the network diagram template for information flow traversing an internetwork consisting of three (3) networks. A complete internetworking diagram only needs to be prepared for a Level 3 diagram. For Level 1 and Level 2 diagrams, only the *user data* field in the internetworking diagram template needs to be included in the diagrams for those levels. In reference to Figure 5-15, the network diagram template has the following attributes:

- **Network Name.** This is the specific NARA "internal" network name for the network, such as the name of a specific departmental LAN. The NARA project manager and project team should consult with NH to identify the proper name of the network, if they are uncertain. For the Internet, indicate the term "Internet" as well as the port, such as For Level 1 and Level 2 diagrams, the collection of network names that the information, data, or transaction must traverse as part of the service path connecting program layers on configured platforms can simply be entered as annotations to the arrow linking the program layers. See Figure 5-8.

- **Network Speed.** This field in the template is the speed at which data, information, or data traverse the network, in Kbps or Mbps. Industry terms may also be used (e.g., T-1, DS-3, OC-3, etc.).

- **User Data.** The primary use of this field is to identify the specific data, information, or transaction to be communicated among program layers via the network. The field should be clearly annotated to indicate such data, information, or transactions as either coming to or from given program layers. The information in this field in the template is summarized at a high-level for the Level 1 and Level 2 diagrams.

  This field can also be used to convey any information the project team would like to document about the network in such areas as LAN operating system, LAN topology, security and access controls, network reliability, etc

- **Transport Layer.** Enter the name of the transport solution (e.g., NetBios, IPX/SPX, TCP, VPN over TCP/IP, etc.)

- **Network Layer.** Enter the name of the network solution (e.g., IP, X.25, NCP, FDDI, ATM, IP over ATM, etc.)

- **Link Layer.** Enter the name of the link layer solution (e.g., 802.x or SDLC)

- **Physical Layer.** Enter the name of the physical layer solution (e.g., RS449, FDDI, RS232, SONET, etc.)

The above attributes define the major characteristics or a network. The definition of transport and network layers is particularly important in designing new IT systems, because they define the type of network upon which the program layers must interact (e.g., X.25 packet switching, SNA, ATM, etc.)

As shown in Figure 5-16, an internetworking diagram is simply an ordered set of network diagrams, where a straight line is used to connect the individual network diagrams. In developing the internetworking diagram, it is not uncommon for a network to encapsulate another network's protocols within it. A virtual private network (VPN) is an excellent example.

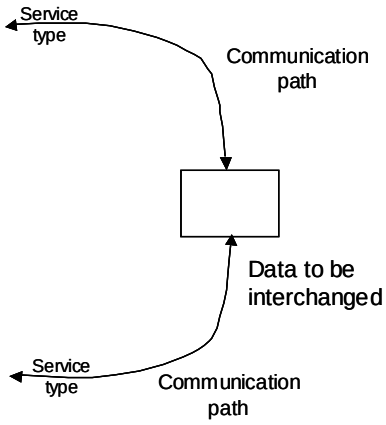
As a rule-of-thumb, if the internetworking diagram becomes too clumsy to draw because of the large number of networks being traversed, draw only the end networks (i.e., the networks that interface directly with the service structures) and the slowest network and the fastest network in between. Use dotted points between the networks to indicate that networks are missing. If part of the internetwork is unknown, draw as many of the networks, as practicable and end the drawing with a dotted line to indicate that networks are missing.

For interoperability between program layers on the same configured platform, the internetworking diagram reduces to a single network diagram. Place the inter-process communication option used (e.g., messages, shared memory, named pipes, etc.) into the network field in the network diagram template.

In preparing Level 3 diagrams, the internetworking diagram may purposefully be drawn a s separate diagram (This makes it be a *part.*) The internetworking diagram may then be pointed to by name by many different middleware boxes that use the same internetworking option.

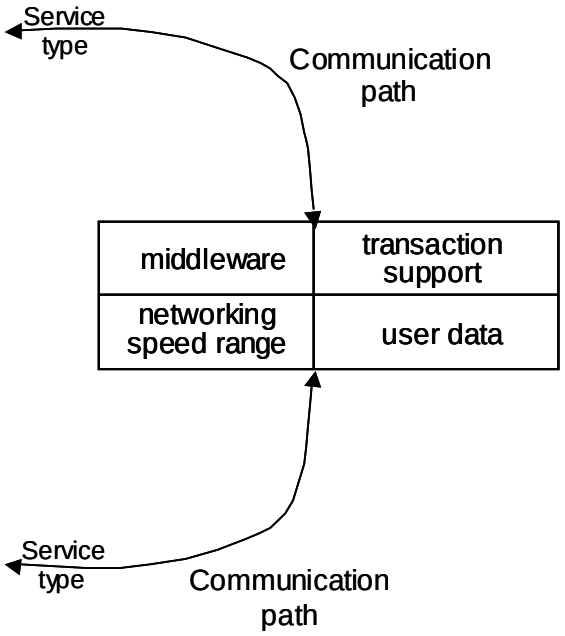
Guidelines for Diagramming Service Paths. In preparing the system engineering diagrams and depicting the service paths, the following summarizes the guidelines for preparing the Level 1 and Level 2 diagrams. Level 3 diagrams should be prepared in accordance with the complete set of instructions defined above.

For **Level 1 diagrams**, the following information should be displayed as annotations to the "simplified" services path template depicted to the right:



- Indicate the service type linking the configured platforms. See Table 5-1 for candidate service types. The reader is reminded that for Level 1 diagrams, the configuration template only refers to the system or product level description and not the detailed program layers.
- In the box in the figure, indicate in bulletized format the type of user data or information to be interchanged between the product and any other NARA systems, with which the product will interface. Clearly indicate whether the information is coming to the product, or being provided by the product to another system. The specification of the user data should be specific enough so that NH and the Architecture Team can assess "what" type data, information, or program controls are required to flow to/from the product to any other interfacing system. Such information is expected to be highly useful in evaluating Product Plans and ConOps Documents for new products to assess any potential operational impact or requirement for re-design or re-engineering of interfacing systems.
- Indicate at a high-level and in a bulletized format, the communications path(s) proposed to be traversed in linking the proposed new product/system with any other interfacing system. The specification of the communications path should be specific enough so that NH and the Architecture Team can assess "what" is the proposed communications path for the new product with any other interfacing systems. Examples might include: Existing departmental LAN, NARANET WAN connections, Internet, virtual private network (VPN), Extranet, etc. The proposed communications path may also be manual. For example, the use of transportable media such as a CD-ROM or floppy diskette.

For **Level 2 diagrams**, the following information should be displayed as annotations to the "simplified" services path template depicted to the right. In the Level 2 diagram, additional information must be provided for the middleware template as shown in Figure 5-14. The reader is reminded that for Level 2 diagrams, all of the specific program layers on the configured platforms for the proposed new product and for any other interfacing systems must be identified and resolved: Guidelines for completing the services paths for a Level 2 diagram include:



- As annotations to the arrows, indicate the service type linking all of the program layers for all configured platforms. See Table 5-1 for candidate service types.
- In the user data field in the middleware box in the figure, indicate in bulletized format the type of user data or information to be interchanged between the given program layer and any other interfacing program layer. Normally, this user data will be more specific, yet derivable from, the user data specification in the services path for a Level 1 drawing. Clearly indicate in the bulletized listing whether the information is coming to or from the given program layer.
- In the middleware field in the middleware box in the figure, indicate the class of middleware that will be used to oversee the connection between the program layers. Table 5-2 provides a detailed listing of the classes of middleware that are generally available.
- In the transaction support field in the middleware box in the figure, indicate "YES" or "NO" whether this interoperability event fully supports transaction processing as discussed above. Criteria for assessing this transaction support are complex and should be discussed with the NH Architecture Team.
- In the networking speed range, indicate the minimum required lowest and recommended speed for the set of networks that transactions will traverse for this interoperability event.
- Indicate at a high-level and in a bulletized format, the communications path(s) proposed to be traversed in linking the program layer with any other interfacing program layer. The specification of the communications path should be specific enough so that NH and the Architecture Team can assess "what" is the proposed communications path for the given program layer with any other program layer. For communications paths linking program layers on the same configured platform, indicate the inter-process communication option proposed to be used on the platform (e.g., messages, shared memory, named pipes, etc.)

6. NARA Configuration Management Guidelines Version 1.0

6.1. Introduction

Information and the systems that make information available to users are critical to the successful operation of the National Archives and Records Administration (NARA). As stated in NARA's Strategic Plan, 1997-2007, we must provide "*ready access to essential evidence"*. Therefore, it is imperative that the agency develops enduring and maintainable information systems, while protecting and managing the components of these systems. In accordance with the Information Technology Management Reform Act (ITMRA) guidance, NARA utilizes commercial-off-the-shelf (COTS) products, whenever possible to develop the major information systems. In addition, NARA acquires systems utilizing contractor support and manages the acquisition with a small cadre of project managers and system engineering professionals. In this regard, configuration management (CM) plays a vital role by controlling the change process to ensure that systems are well documented, functioning and maintainable.

CM can be loosely defined as a management discipline of initiating, evaluating and controlling change of the project components during and after the development process[1]. During the IT system development phase, the CM manager works closely with the project manger and *initiates* identifying the components of the software, hardware and documents NARA wants to track and baselines them in the CM tools. During this phase, the CM manager ensures that the contractor maintains CM activities using their own tools with the NARA-approved CM standards and procedures. However, after the final product is delivered and deployed, NARA assumes the CM responsibilities. During this operation phase, CM activities focus on *evaluating and controlling* the proposed changes to the product. CM must be properly implemented during all phases of a project to ensure that modifications and enhancements to systems are defined and managed. This will ensure that systems under development are delivered within cost and schedule, and that operational systems continue to function as upgrades occur.

6.1.1. Purpose of the Document

This document describes the CM process for all NARA IT systems and roles and responsibilities related to the process. For clarity, the document addresses CM in two phases. The first is the role of CM during IT systems development, and the second refers to CM during operations and maintenance.

During the development phase, the focus of CM is on managing CM requirements and baseline design. Changes to the approved baseline are usually initiated as a result of a design change or functional requirement change while customizing a COTS product. During this phase, the CM manager would work closely with the Project Manager in charge of the development effort. During the operations and maintenance phase, the focus is on keeping the system operational.

[1] CM definition expanded and modified from SEI-86-TR-5, Summary of the SEI Workshop on Software Configuration Management, Software Engineering Institute, Carnegie Mellon University, December 1986

Therefore, change requests are most often generated by the end-users. Functional changes which represent enhancements to the system would normally be grouped, managed as a new project, and then delivered to operations as a new product release. During this phase, the CM manager would work closely with the Product Owner.

6.1.2. Scope

CM identifies and controls the software specifications as well as the hardware components and COTS products throughout the product lifecycle. Since the hardware and COTS product specifications rarely change except for upgrade, CM will identify these as very high level components. On the other hand, CM will require well-defined documents of the developed or configured software components. NARA acknowledges that adequate documentation is critical to building a maintainable application. In addition, CM will maintain the records of the proposed changes to the approved documents, and facilitate communication by providing status of the changes to all parties involved.

6.1.3. Recommended Approach

Proposed changes to IT systems will follow a three tier-approval process consisting of the Project Manager (PM)/Product Owner (PO), the Configuration Control Board (CCB), and the Investment Review Board (IRT). The table below provides a general guide for the approval of changes. The change request will include a justification, analysis and impact statement of the change. All change request recommendations and/or approvals will be signed by the appropriate authority. The detailed roles, responsibilities and processes are defined in the following sections of the document.

Type of Change Request	Recommend	Approve
Contractor or Government initiated; No impact to underlying technical design AND Does not cross functional areas AND No impact to cost AND No impact to schedule	PM/PO	PM/PO
Contractor or Government initiated; Impacts to underlying technical design or Crosses functional areas AND No impact to cost AND No impact to schedule	PM/PO	CCB
Contractor or Government initiated; Impacts to underlying technical design and no CCB consensus OR Crosses functional areas and no CCB consensus OR Impact to cost OR Impact to schedule	CCB	IRT

6.2. Roles and Responsibilities

6.2.1. NARA CM Manager

The NARA Configuration Manager is responsible for developing and maintaining the agency-wide CM processes and procedures. The CM manager will work closely with the PM during IT development phase and with the PO after the deployment of the system to ensure that CM processes are established and adhered to for the lifecycle of the IT system. The system lifecycle begins with concept development and ends with formal retirement of the production system.

The CM manager is the Chairperson of the NARA Configuration Control Board. In that role, the CM Manager or designee will conduct the CCB meeting, publish agendas and minutes, and approve the change requests with consultation of the members or submit them to Investment Review Team (IRT) for approval. The CM manager is also responsible for communicating all decisions at all levels to the IRT.

6.2.2. NARA Configuration Control Board

The NARA Configuration Control Board (CCB) approves CM processes and procedures and is the CM implementation group for all cross-office projects. The CCB reviews all CM activities and assists the CM manager, as required.

The CCB is also responsible for reviewing and approving all change requests submitted by the PM or PO. Besides the CM manager, standing members of the board include the Software Architect, the Network Architect, Director of NHT, the Data Administrator and the Data Base Administrator. Depending on the nature of the changes, the CM manager will also identify any other technical members of the CCB that may be asked to advise based on their area of expertise such as, testing or quality assurance.

For a specific IT development project, the PM will be a standing member of the CCB. The PM will also identify the other functional representatives for the CCB. These members should collectively represent all of the functional areas described within the Baseline Requirements Document. The goal is to obtain representation from all affected offices. For the Operations and Maintenance Phase, the PO will have this responsibility.

The CCB holds a regular meeting once a month or can meet at the request of the project manager, the product owner or the CCB members.

6.2.3. Project Manager (PM)

The Project Manager is responsible for ensuring that the Contractor CM process is acceptable and all work performed by the contractors is properly placed under their CM process prior to delivery. When the project manager approves the work of the contractors, the government CM manager baselines the deliverables under the government CM process. Specific baseline deliverables include requirements documents, design documents, test reports, and product codes. Without establishing the baseline at each phase, the contractor is not authorized to move on to the next phase of development.

The PM will manage the baseline changes related to the project with support of the Contractor's CM process during the development phase. The PM will approve the Baseline Change Requests (BCRs) with consultation of the project team or submit the BCR to the CCB for approval. The PM will work with the CM manager to ensure that change requests are scheduled for discussion at the CCB meeting, as needed.

The PM is also responsible for the transition of Contractor delivered products to the Government to formally begin the Operations and Maintenance Phase.

6.2.4. Product Owner (PO)

The PO will work closely with the CM Manager at project transition to ensure that all Baseline products, deliverables and documentation are complete and are placed under the Government CM process. The PO will also work with the operations staff to ensure that the system is meeting user expectations.

The PO will manage all System Change Requests (SCRs) related to the product with support of the Government's CM process. The PO will approve SCRs with consultation of the CM manager and operations staff or submit the SCR to CCB for approval. The PO will work with the CM manager to ensure that change requests are scheduled for discussion at the CCB meeting, as needed.

The PO will also recommend, based on the number and complexity of SCRs, when a new project should be initiated to develop a new release of a product. This action will begin with the development of a new or updated Product Plan.

6.2.5. Investment Review Team (IRT)

CIO, as a member of the IRT, is responsible for approving the membership of the NARA CCB. Based on recommendations of the CCB, the IRT will have final approval of all change requests, as required.

6.3. CM During IT Systems Development

In the NARA IT development environment, there is a clear distinction between the CM activities during the development phase and during the operation phase. During the IT development phase, the government CM focuses on managing CM requirements and baselines. During this phase, CM activities are mostly performed by the contractors using their own tools. After the product is deployed and operational, the Government assumes the responsibilities of CM activities. During the operation phase, CM activities center on managing the change requests generated by the product end users. CM during the development phase is discussed below, while CM during the operation phase in the following section.

6.3.1. Configuration Item Identification

A configuration item (CI) is an item we wish to track in CM activities. Identifying configuration items is one of the most important steps to a maintainable project development effort. It determines the level, amount of documentation required, the amount of testing, and the ease of maintenance. During the project initiation phase, the PM and the CM manager develop a list of high level functional requirements, and show how they are satisfied by the hardware, software and the COTS components.

6.3.1.1. Configuration Items

Identifying CIs is a top-down decomposition process of dividing software, hardware and COTS components into a logically related hierarchy of parts. This decomposition process continues down to the lowest logical elements during the development phase. Since the hardware and COTS product specifications rarely change except for upgrade, CM will identify these as very high level components. On the other hand, CM will require well-defined documents of the developed or configured software components. The PM and the CM manager identify CIs and assign the identification number to them using the NARA Standard CI Numbering System (see Appendix A).

6.3.1.2. Required Documents

In addition to identifying the SW, HW and COTS components, required documents and document standards must be identified and discussed with the PM and contractors during the project initiation phase. A project specific waiver must be attached for a waived document. Adequate documentation in a well-defined format is necessary for building a functional product. The PM should ensure that all documents are prepared adequately during the development phase, and provide definition document and acceptance criteria document for each CI.

6.3.2. Baseline Design

During the development phase, the project components are constantly changing. It is unnecessary, however, to attempt to control all changes. Configuration identification is based on the concept of baseline management. A baseline is an approved plan and becomes the standards against which all changes are measured. The CM manager is required to establish the baseline at the end of the requirements, detailed design and deployment & acceptance phase, and provide the baseline reports for the PM, PO, CCB and IRT.

6.3.2.1. Functional Baseline (FBL)

At the end of the requirements phase, the PM conducts the Requirements Review (RR), and delivers the following requirements documents to the CM manager. The CM manager verifies that all required documents and configuration items are established properly in the definition documents. They will be established by the CM manager as a FBL.

- Concept of Operations
- Requirements documents
- Analysis of Alternatives

6.3.2.2. Allocated Baseline (ABL)

At the end of the design phase, the PM conducts the Critical Design Review (CDR). Upon the approval of the detail system design, the PM and the contractor are required to deliver the following design documents to the CM manager. The CM manager verifies that all required documents and configuration items are established properly in the definition documents. Then the CM manager establishes them as an ABL.

- Preliminary System Design documents
- Detailed System Design documents
- Developer Test Plan

6.3.2.3. Product Baseline (PBL)

At the end of the Deployment & Acceptance phase, the PM conducts the Acceptance Review & Audit (ARA) meeting. The Government Quality Assurance (QA) representative (or PM) must verify that the results of the acceptance test meet or exceed the stated acceptance criteria. Upon the approval of the delivered product, the PM and the contractor are required to deliver the following items with the QA certificate to the CM manager. The CM manager verifies that the tested product is exactly described in the PBL definition documents. They will be established as a PBL.

- Developer Test Report

- System Test Plan/Report
- Integration Plan
- Acceptance Test Plan/Report
- Installation and Transition Plan
- Training Plan
- Training Material
- Operations Manual
- Users Manual
- Configuration Items (Product Codes)
- Configuration Management data (archive file)
- Version Description Document
- *All contractual deliverables.*

6.3.3. Baseline Change Procedure

During the development phase, if the need to change the baseline arises, the contractor is required to submit the baseline change request to the PM. In general, the SCR resolution process in section 4.2 applies to the baseline change request as well. If the proposed change does not impact underlying technical design, cross-functional area, cost or schedule, the PM can approve the change. Otherwise, the PM must submit the change request to the CCB for approval. If the baseline change increases cost or affects the schedule significantly or if the CCB cannot reach a consensus among members, the CM manager must submit it to the IRT for approval.

6.3.4. Contractor CM requirements

During the development phase, contractors are required to perform and document CM activities on their CM tools at their sites. After the system acceptance testing, the PM approves the product at the Acceptance Review & Audit meeting (ARA). The contractors are required to deliver product codes, and associated documents, CM activities tracking document, and all contractual documents *in the proper format and media* so that the CM manager can import or copy to the NARA CM tools. The hot fixes or the problems reported during the final stage of the testing or deployment must be documented and submitted to the CM manager. The contractor's CM standards and procedures must be approved by the Government CM manager at the project initiation phase. The Government CM manager can inspect the contractor's CM activities at any time.

6.3.5. CCB Approval requirements

The baselines (FBL, ABL, and PBL) will be established by the CM manager and must be approved by CCB before moving to the next phase of the development or deployment. The PM may invite the CM members or designated technical staff to the review meetings or arrange

separate meetings with CCB for approval. The CM manager provides the CCB approval notice to the PM.

6.4. CM During The Operation Phase

After the product is deployed and operational, CM activities center on managing the change requests generated by the product end users. The process of change control is the most critical area of CM. It requires a systematic evaluation, coordination, and approval/disapproval of proposed changes, and the implementation of approved changes to the established baselines.

6.4.1. Change Request Procedure

During the operation phase, all problems should be reported to the NARA Help Desk. The Help Desk staff sorts out the problems received and reports to the product owner (PO). The PO evaluates the problems and impacts with the assistance of the technical staffs. Automated CM tools will be employed to track the change requests.

6.4.2. Change Request Resolution Procedure

As stated in the section *Approach*, proposed changes to IT systems will follow a three tier-approval process consisting of the Product Owner (PO)/Project Manager (PM), the Configuration Control Board (CCB), and the Investment Review Board (IRT). Proposed change requests should be examined to see how the change would affect the following categories. Depending on the result, the approval authority will be different.

(1) Impact to underlying technical design

Underlying technical design change includes:

- Addition of a new table
- Addition of a new column to the existing table
- Data type change of the existing column
- Alteration of the primary key or unique key
- Change in the stored procedure that is commonly used in many modules
- Index change of the frequently used columns
- Revision of the view structure
- Interface program change within the system
- A major change in screen design.
- A major change in user or operator's manual
- Specification change in hardware.

(1) Cross functional area

The change in the program or module that interacts with the other system.

(3) Impact to cost
Significant change in cost. It will be further defined in the project CM plan.

(2) Impact to schedule
Significant change in schedule. It will be further defined in the project CM plan.

If there is no impact to all the above categories, the PM/PO can approve the change with the consultation of a project team or technical staff. However, if proposed change requires modification in the underlying technical design or cross-functional area, or a significant change in cost or schedule, the PM/PO must submit the system change request (SCR) with impact analysis to the CCB for approval. If the CCB cannot reach a consensus among members or the proposed change request significantly impacts cost or schedule, the CCB must submit it to the IRT for approval (See section 1.4 table). For a mission-critical change request that needs immediate fix, the PO must consult with the CM manager for a quick resolution. See Change Request/Resolution Flowchart in Appendix G.

6.4.3. Code Control Procedure

6.4.3.1. Code Change Procedure

Once the change request is approved, the technical staff who fixes the problem is required to document the change information. They will then check out the affected module from the Government CM tools in order to fix it, then check it in after a successful test. All documentation should follow the agency's documentation standards.

6.4.3.2. Release Procedure

When the fixed codes or documents are ready for release, the CM manager shall prepare the Version Description Document (see Appendix E). The technical staff will release the affected modules with VDD to the affected sites. The release can be scheduled on a regular basis (e.g., BI-weekly) or immediately for mission-critical change request. The release schedule and conflict resolution will be further defined in the project CM plan.

6.4.4. CM Status Report

The CM manager should provide the PM, PO, CCB, IRT and the end users with the status report related to the baselines, release/versions, and the changes pending to them. The frequency of the report should be discussed at the project initiation phase. Most of information should be retrieved from the automatic CM tools. In addition to the baseline status report and release report mentioned above, the status reports include:

- Change request status report
- Deferred change request status report
- CCB meeting minutes
- *Ad hoc report.*

6.4.5. CM Functions by Participants

This table shows the matrix of the major CM functions and the principal participants.

PHASE	FUNCTIONS	CM mgr	PM	Cntrctr	PO	CCB	User	Tech	IRT
Developmnt	Identify CI & req. Doc	X	X						
	Baseline Design	X	X	X					
	Development CM			X					
Operation	Problem Report						X		
	Change Resolution				X	X		X	X
	Change Release	X						X	
	Status Report	X							

6.5. CM Support Functions

6.5.1. Library Support

All CM related documents, diagrams, product codes, reports, meeting minutes, and tracking information are required to be stored in the software library at the agency-designated site, and to be maintained by the NARA staff. During the maintenance phase, designated contractor staff may be allowed to check in/out the affected modules.

NARA staff is required to back up the updated information on a weekly basis. CM archive copies must be stored at another site. Only the CM manager and designated staff can have access to the archive. CM information shall be transferred from the agency software library to the archive on a monthly basis.

6.5.2. CM Tools

Managing CM activities is a complicated task. It requires the agency to utilize CM tools to manage CM activities such as baselining, change control, and tracking of the change requests. During the development phase, contractors will manage CM activities using their own CM tools. However, upon delivery of the product, NARA will assume the responsibility of baselining the final product and managing CM thereafter.

For all change requests CM tools should be available to track its submitter, date/time, current status, and approval status, and it should be able to trace back the version number and associated documents of the fix. It should also provide the status report for the interested parties.

6.6. CM in the Context of the Development Life Cycle

Since CM activities are performed throughout the application life cycle, they can be better understood in the context of the development life cycle. During the development phase, initially the CM manager and the PM are required to establish the project CM plan, identify system requirements and configuration items; The CM manager establishes Functional Baseline (FBL) at the end of Requirements Review (RR). After the PM reviews and approves the contractor's detail design documents via the Critical Design Review (CDR), the CM manager establishes the Allocated Baseline (ABL). After the system development and testing, the PM approves the product via Acceptance Review & Audit (ARA) meeting. The CM manager then establishes a Product Baseline (PBL). During the system development and testing phase, the contractors will perform CM activities at their site using their own CM tools. Upon deployment of the system, NARA will assume the CM responsibilities such as code control, release control and version control. The table below shows CM activities in the context of the Waterfall development life cycle.

Project Initiation Phase		**Development Phase**					**Operation Phase**	
Concept development	Requirement Definition	Preliminary Design	Detailed Design	Development	Integration & System Test	Deployment & Acceptance	Production	Retirement & Rollover
Concept Review	Requirement Review	Preliminary Design Review	Critical Design Review	Test readiness Review	Production Readiness Review	Acceptance Review & Audit	Retirement Rollover Review	
CR	RR	PDR	CDR	TR	PRR	ARA	RRR	
CM plan. & CI	Functional Baseline (FBL)		Allocated Baseline (ABL)	*Contractor performs CM activities*		Production Baseline (PBL)	System Change Request (SCR)	

6.7. Contractor/Subcontractor/Vendor Requirements

All contractors or subcontractors are required to submit their CM standards and procedures to the NARA CM manager for approval at the project initiation phase. The contractor's CM standards and tools must be comparable to those of NARA, and the documents and contractual deliverables should be imported or copied readily to the Government CM tools. The contractors

are encouraged to use the NARA Standard CI Numbering System while working for the NARA project.

6.8. References

- Carnegie Mellon University, Software Engineering Institute, Configuration Management Plans: The Beginning to your CM Solution.
- IEEE Standards 1042-1987, IEEE Guide to Software Configuration Management, 1999.
- U.S. Department of State, Software Engineering Standards and Procedures, Volume III. Carnegie Software Configuration Management, November 1993.
- U.S. Department of Commerce, CM-001-1993, Configuration Management Plan, October 1993
- *Principles of Configuration Management*, M.A. Daniels, Advanced Applications Consultants, Inc., 1985

6.9. Appendices

A. NARA Standard CI Numbering System
B. Acronym
C. Glossary
D. Version Description Document
E. Change Request Form
F. Change Request/Resolution Flowchart

A. NARA Standard CI Numbering System

A.1 Configuration Item Numbering System

In NARA, each major system will be assigned a uniquely identifiable configuration item (CI) number. The CI number will then be expanded on as the system is broken down into components (modules, parts) during the development life cycle. The CI numbering system will be used to identify application software, hardware and DBMS, COTS products, and associated documents.

A.1.1 The Format of the CI Number

The CI numbering system is a three-part identification in the format of **SYSTEM TYPE-PROJECT-CI Type-CI Subtype.XXX** where:

- SYSTEM TYPE is a three-character field that indicates the type of system that is being developed. Current possibilities are:

  ADM for Administrative Application
  RLC for Record Lifecycle Applications
  SML for Small system

- PROJECT is a seven-character field that contains the agency's acronym for the subsystem (i.e. OFAS, ARC etc). The last two digits indicate the subsystem development phase, with 00 as the first phase (e.g. OFAS first phase is OFAS00, and the next phase is OFAS01).

- CI type is a two character field, where the characters are either
  HW for hardware and RDBMS,
  SW for software or customized component of software,
  CT for COTS product (not customized) or
  DC for document (e.g. module description).
- CI subtype is a three-character field.

CI Type	CI subtype	Meaning
HW	SRV	Server
	DBM	DBMS
DC	PLN	Planning phase
	DSN	Design phase
	OPR	Operational phase
	TST	Test phase
	PRT	Prototype phase
SW	EXE	Executable
	RPT	Report module
	SCR	Screen module
	WEB	Web module
CT	GPL	GreatPlains (example)

- XXX represents a three-digit identifier used to identify CIs at the lowest level, usually represents an actual program file or document. For document, use pre-defined document code for the lowest document file.

A.1.2 Examples of the CI number

(1) Software components

RLC-OFAS00-SW indicates the OFAS subsystem software component as a whole for the first phase.

SCREENS will be tracked with SCR subtype:

RLC-OFAS00-SW-SCR.101	Form80 Order Entry screen
RLC-OFAS00-SW-SCR.102	Customer Maintenance screen

REPORTS will be traced with RPT subtype:

RLC-OFAS00-SW-RPT.101	Average Processing Time report
RLC-OFAS00-SW-RPT.102	Sales billing report

(2) COTS product components

RLC-OFAS00-CT	OFAS subsystem COTS component as a whole.
RLC-OFAS00-CT-GPL	GreatPlains
RLC-OFAS00-CT-GPL.101	GreatPlains User Manual Ver 1.0

(3) Hardware components

RLC-OFAS00-HW	OFAS subsystem hardware component as a whole.
RLC-OFAS00-HW-SRV.101	Citrix.
RLC-OFAS00-HW-DBM.101	MS SQL Server Ver 6.0 .

(4) Document components

RLC-OFAS00-DC	OFAS subsystem hardware component as a whole.
RLC-OFAS00-DC-PLN.101	Requirement Analysis Package.
RLC-OFAS00-DC-OPR.101	OFAS User manual.

As the CM manager determines other components for tracking, identification numbers will be applied in numeric order.

A.2 Release/Version Numbering System

All components will be labeled with the release and version number as they go through the changes. These numbers change dynamically throughout the project life cycle, therefore they are not embedded as part of the configuration identification number.

The release and version number will be the format of **Rx.Vaa.bb** where:

- **R** identifies the release
- **x** is the baseline life cycle phase where
  - 1 identifies FBL
  - 2 identifies ABL
  - 3 identifies PBL
- **V** identifies the version

- **aa** indicates the number of changes to that component within the indicated life cycle phase and stage. It is increased by one with each successive change, and is reset to 01 when a new release is defined.

- **bb** indicates the number of reiterations of a specific version. It is increased by one with each successive version of a given change, and is reset to 01 with each new change. This number tracks multiple versions of a configuration element that are produced as part of resolution of a group of related change requests.

For example:

R2.V03.04 indicates that the document was Allocated baseline (2), that it is the third version (03) and that four revisions (04) have been made to the third version.

B. Acronyms

ABL Allocated Baseline. It is an approved plan at the end of the design phase.
ARA Acceptance Review & Audit (ARA)
ASP Application Service Provider. After the development phase is over, the ASP maintains the system and provides technical evaluation for the proposed change.
BCR Baseline Change Request. When the proposed change request requires the change in the approved Functional and Allocated Baselines, the contractors submit the BCR.
CCB Configuration Control Board. CCB is a CM policy and implementation group. It is primarily responsible for CM activities.
CDR Critical Design Review
CI Configuration Item. The item that NARA wants to track in CM activities.
CM Configuration Management.
FCA Functional Configuration Audit
FBL Functional Baseline. An approved plan at the end of analysis phase.
IRT Investment Review Team
ITMRA Information Technology Management Reform Act
NARA National Archives and Records Administration.
PBL Production Baseline. An approved plan at the end of acceptance test phase.
PCA Physical Configuration Audit
PM Project Management or project manager
PO Product Owner. After the system is operational, the PO represents the end user for the product and is responsible for the operation of the system. The PO evaluates the system change requests, and approves or submits them to the CCB.
QA Quality Assurance
RR Requirements Review
SCR System Change Request.

C. Glossary

Allocated Baseline, approved configuration items at the end of the design phase.
Baseline Change request, a change request that affects the approved functional or allocated baseline item.
CM plan, a definition of CM policies and procedures for a particular project
Configuration, the complete technical description required to build, test, operate and support the product.
Configuration Control Board, a group of technical and administrative personnel who establishes CM standards and procedure, and administers CM activities.
Configuration Item Numbering System, a unique identifier for a CI, composed of system type-project-CI type-CI Subtype.XXX.
Configuration Item, a collection of hardware, software or COTS components that satisfy the requirements of the product.
Functional Baseline, approved configuration items at the end of requirements phase.
Functional Configuration Audit, a formal examination of CI's functions to verify that the item meets the functional requirement specified in its documentation.
Interface, a common module or program that connects to each other between two or more modules.
Operational, applies to actual use of a product.
Physical Configuration Audit, a formal audit to verify that an "as built" CI conforms to the documentation.
Product Baseline, approved configuration items at the end of development/test phase.
Prototype, the first model of a product which demonstrates the functional and physical requirements.
Quality Assurance, the system, procedures and activities for assuring that an item performs according to the specification.
Revision, a modification of a program or document.
System Change Request, a change request that affects the product baseline.

D. Version Description Document(Annotated Outline)2

Title Page
Preface

Table of Contents

1.0 Introduction

.1 Purpose
The VDD provides information and guidance for installation and using a specific release. Include the complete baseline identification of the new baseline and the baseline being replaced.

.2 Scope
Describe the functional and physical changes associated with this release.

.3 Definitions
Provide new definitions unique to this release. Reference other standard definitions in other documents.

.4 References
(As needed)

2.0 Functional Description

.1 Capabilities
Provide a summary of deleted, added or significantly modified functions or capabilities. Reference to the document for more information located.

.2 Documentation
Provide a summary of significant changes to the documentation such as user or operator manuals.

3.0 User Considerations

.1 Special processes
Describe special actions the user must take to use the new release. (e.g. changed function keys, procedure etc). Include reference to other documents for more information.

.2 Incorporated Changes
Identify the SCR that has been incorporated in this release. This section can be presented in the form of a modified baseline status report.

.3 Deferred changes
Identify the SCR that has NOT been incorporated in this release.

4.0 Inventory
Identify the physical media that contain the release. List the CIs.

5.0 Installation Instructions

Provide installation instructions that are site specific and detailed. Include procedure for ensuring that installation was accomplished correctly.

[2] Copied from VDD U.S. Department of State, Software Engineering Standards and Procedures, Vol. III SCM, November 1993.

E. System Change Request Form

System Change Request

PROBLEM REPORTER INFORMATION Attach any referenced material or relevant information

PROJECT_____ **CONTROL NUMBER**_____

Report Date_____ **Time**_____

Name_____ **Office**_____ **Phone**_____

PROBLEM SUMMARY

Type of Change:
Software_____ **Hardware**_____ **Document**_____ **Other(*Specify*)**_____

Priority: **Critical**_____ **Urgent**_____ **Routine**_____ **Low**_____

PROBLEM DESCRIPTION

Software/Hardware/Document Affected (e.g. Screen or Report Name):

Problem Description (e.g. Input, Expected Results, Anomalies, or Reference to manual, etc.):

TECHNICAL EVALUATION by Technical Staff

Describe detail technical/impact analysis (e.g. interface, cost, schedule or design etc.)

Impact Analysis Summary **CCB approval required if change impact to the category below is significant**

Check the categories proposed change would affect: **No impact to the below**_____
Technical Design_____ **Cross-functional Area** _____
Significant Cost_____($) **Significant Schedule change**_____(wks/months)

FOR APPROVING OFFICER ONLY **Product owner if there is no impact, otherwise CM manager**

Approved_____ **Approved for future upgrade**_____ **Disapproved**_____

Name_____ **Office**_____

Signature_____ **Date**_____

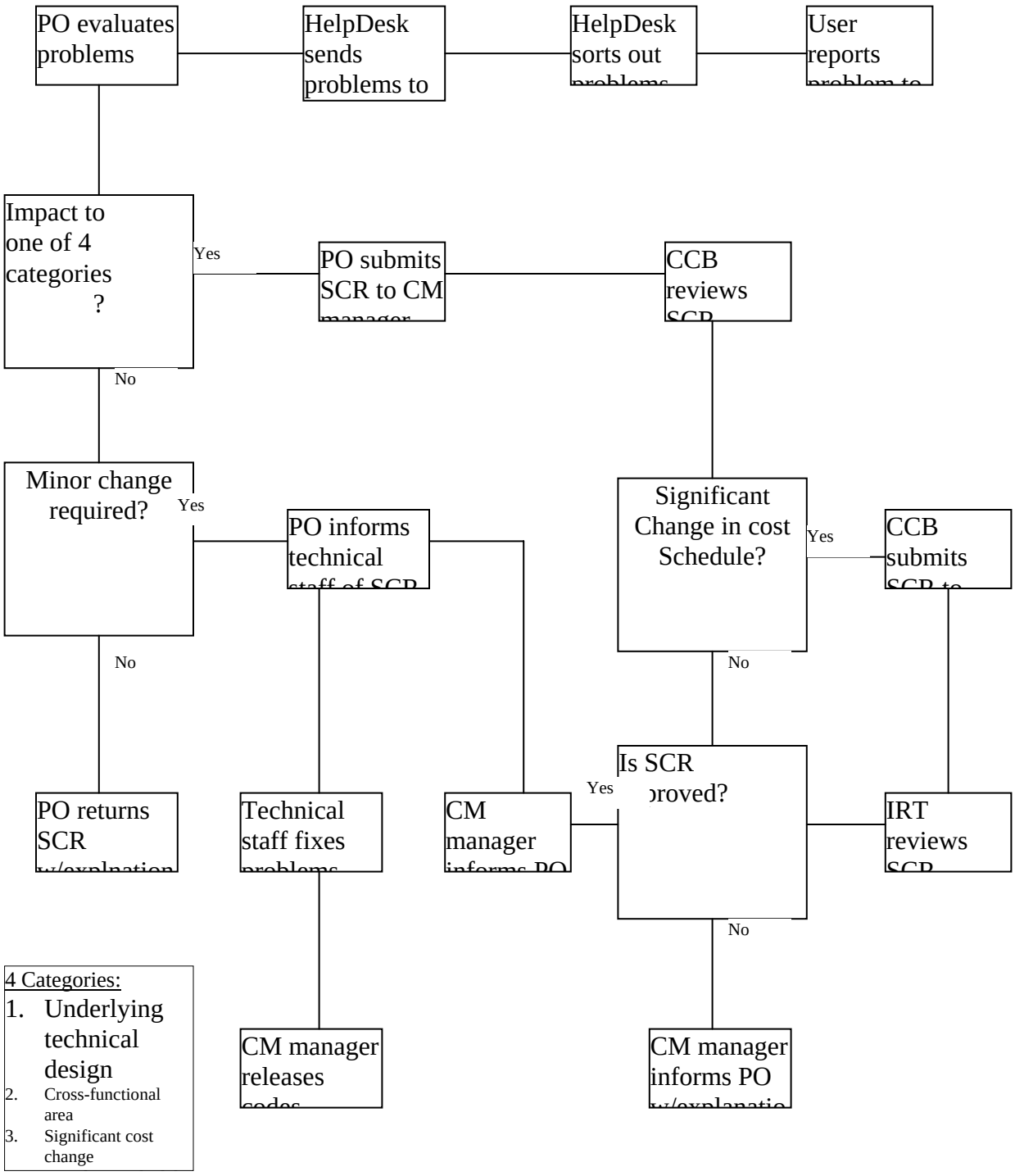
Reasons for disapproval(*Use the back space or attach a separate sheet if necessary***):**

TASK ASSIGNED

Contractor or NARA Office:	Assigned To:	Resolution Date:

Contractor or NARA Office:	Assigned To:	Resolution Date:

F. Problem Report/Change Request Flow Chart



7. IT Security in the Life-Cycle Process

7.1. Overview

Security is vital to preserving NARA's investments in IT resources. Since it is not possible to consider and apply security concepts and measures randomly and still maintain an acceptable level of security, security should be included throughout the life-cycle process, from inception through retirement from service. Much of the guidance comes from recommendations of the National Institute of Standards and Technology (NIST), specifically from Special Publication 800-64 which provides additional information on security considerations in the Information System Development Lifecycle. Although the model described is generic and may not follow exactly the life-cycle process used by all organizations, this model demonstrates that every milestone in the life-cycle process has a corresponding security component.

7.2. The Life-Cycle Process

The life-cycle process spans the entire time that a project/ program is being planned, designed, developed, procured, installed, used, and retired from service.

7.2.1. Security Responsibility in the Life-Cycle Process

The project/program manager is responsible for ensuring that security concerns are included in the life-cycle planning process. Although the manager has this responsibility, other staff members involved in developing, operating, and maintaining systems are responsible for factoring security concerns into the work they do and the decisions they make. These staff members include application developers, system designers, System Administrators, application managers, and Information Security Officers

7.2.2. Relationship Between the Life-Cycle Process and IT Security Planning

IT security planning is closely aligned to the life cycle of a project or program. Many of the tasks described in this paragraph correlate to IT security planning tasks (such as assessing risk and selecting controls). When a security task has a related task in IT security planning, a cross-reference to the appropriate IT security planning paragraph is provided. The paragraphs for IT security planning describe the way to complete some of the key security tasks in this section

7.2.3. The Life-Cycle Process Phases

Each phase consists of a set of project tasks. For each project task, a corresponding security task should be completed. The paragraphs that follow describe both the tasks for each phase in the life cycle of a project and the corresponding security tasks that should be taken. The life-cycle process consists of the following phases:

a. Concept Development

b. Requirements Definition.

c. Design.

d. Development.

e. Installation, integration, and testing.

f. Deployment and Acceptance.

g. Production and Upgrades.

h. Retirement and Rollover (Disposal of assets).

7.2.4. Concept development

Project Task	Corresponding Security Task
Identify user needs: • Identify mission, resources, and priority. • Describe basic requirements and objectives. • Provide a general statement concerning the nature of the service requested and overall concept.	Identify security needs: • Identify the kinds and sensitivity of information that will be stored, processed, or transmitted. (See paragraphs 1.2.8 and 1.2.9 for more information.) • Identify basic security objectives and goals. • Identify the security resources needed to manage security. • Preliminary Risk Assessment (See NIST 800-30, *Risk Management Guide for Information Technology Systems*). • Security Categorization (See FIPS 199, *Standards for Security Categorization of Federal Information and Information Systems*).

Project Task	Corresponding Security Task
Evaluate alternatives to satisfy requirements: • Identify alternatives to satisfy requirements. • Analyze technical, operational, and economic feasibility. • Consider costs versus benefits.	Identify security alternatives for each requirement by performing an initial risk assessment: • Identify threats, vulnerabilities, and risks. • Analyze technical, operational, and procedural controls for their economic feasibility as project security alternatives. • Estimate security-related costs versus benefits including cost considerations and reporting (See NIST 800-55, *Security Metrics Guide for Information Technology Systems*). See paragraph 1.2.10 for guidance on completing these steps.
Select and approve one approach, establish project objectives, and provide a general definition of the requirement and system or application architecture.	Identify the basic security framework in the selected alternative and provide essential vulnerability information with impact issues and concerns.

7.2.5. Requirements Definition

Project Task	Corresponding Security Task
Prepare a project plan to guide the development effort, budget, and schedules. Include methods for design, coding, documentation, problem reporting, and configuration management in addition to verification, validation, and testing.	Develop Security Plan (See NIST SP 800-18, *Guide for Developing Security Plans for Information Technology Systems*) to include security checkpoints in the project plan. These checkpoints include quality assurance for development of security controls, identification of configuration and change control process, and development of an internal audit plan.
Develop a basis for design from user requirements (i.e., functional requirements).	Define security requirements by doing the following: • Risk Assessment and Risk Mitigation (e.g., Identify new threats, vulnerabilities, and risks). • Identify user requirements for protecting data. • Determine where security requirements apply in the system. • Determine the mix of security controls. The appendix lists the baseline requirements that the system or application should meet including Security Function and Assurance Requirements (e.g., see NIST SP 800-23, *Guide to Federal Organizations on Security Assurance and Acquisition/Use of Tested/Evaluated Products*). Develop recommended Security Controls for System (See NIST SP 800-53, *Recommended Security Controls for Federal Information Systems*).
Develop preliminary test plans.	Refine the risk analysis and develop a contingency plan. Develop preliminary security test plans that do the following: • Describe the security objectives (i.e., goals), policies, and requirements. • Identify the resources needed to test the plan and establish a test schedule. • Describe evaluation criteria and areas to be tested.

Project Task	Corresponding Security Task
Select an acquisition strategy that is commensurate with cost, risk, and need.	Design requests for proposals (RFP's) and contracts to include security requirements. Design solicitations to provide quality assurance of security controls. Evaluate offers for addressing the adequacy of security controls.
Update requirements analysis and formalize the requirements into a functional baseline.	Include security requirements in the formal functional baseline.

7.2.6. Design

Project Task	Corresponding Security Task
Develop a detailed design and supporting specifications for the project, including requirements for input, output, files, databases, and system controls.	Define security specifications, including identification of the following: • System/subsystem and interface security specifications. • Program, database, hardware/firmware, and network security specifications.
Develop and update verification, validation, and testing goals and plans.	Update the security test plan and develop a security test procedure. Determine that the procedure will test security specifications and performance under both normal and abnormal circumstances.
Design a solution that satisfies requirements and constraints, and establish a formal functional baseline.	Include security specifications in the formal functional baseline.

7.2.7. Development

Project Task	Corresponding Security Task
Construct the system or application from detailed design specifications.	• Develop security code, control access to security code, and identify and document security code, as appropriate to the project. • Develop Security Control Integration. • Develop Security Integration and Acceptance. • Develop Security Certification & Accreditation
Perform unit tests and evaluate the results.	Perform unit tests and evaluate security-related code, as appropriate to the project.
Implement a detailed design that results in a system ready for installation. Establish as the formal developmental baseline.	Include approved security components in the formal developmental baseline.

7.2.8. Installation, Integration, and Testing

Project Task	Corresponding Security Task
Test system components.	Conduct tests of security in the configured components.

Project Task	Corresponding Security Task
Validate system performance.	Conduct security tests in the integrated system, including assessing functional operation and performance and identifying any test failures. Analyze test results against security requirements.
Install the system with any necessary code modifications.	Install security code with any necessary code modifications, as appropriate to the project.
Prepare users' guides and operations/maintenance manuals.	Prepare documentation of security controls for users' guides and operations/maintenance manuals.
Perform an acceptance test and validate the results.	Conduct an acceptance test and evaluate project security. This testing includes the following: • An assessment of the functional operation, performance, and resistance to penetrations. • Identification of the level and type of security controls. (See paragraph 4.6 for information on penetration testing.)
Accept the system and establish a formal product baseline.	Verify the project security, identify strengths and limitations, and prepare the required IT Security Plan and supporting documentation. (See paragraph 5.1 for the organization and content of an IT Security Plan.) Establish Configuration Management and Control.

7.2.9. Deployment and Acceptance

The following security tasks are part of the IT security metrics. There are security tasks to be accomplished before the system goes into operation and during the operational phase.

7.2.9.1. Before the Operational Phase

Before the system goes into operation, the project manager will ensure that the following have been done:

a. A line manager is assigned responsibility for the IT security of the system.

b. An IT Security Plan is written (see paragraph 5.1 for the organization and content of this plan).

c. The system has been authorized in writing and approved for processing, which signifies the project manager's acceptance of risk.

7.2.9.2. During the Operational Phase

During the operational phase, which could span months or years, the line manager will do the following:

a. Ensure that security is carried out as described in the IT Security Plan.

b. Continually monitor and perform self assessments.

c. Review the security controls for the system or application at least every 3 years or upon significant change to the system, whichever comes first.

d. Re-authorize processing before the operation of the system or application can continue. This re-authorization signifies the line manager's acceptance of risk.

e. Establish metrics to measure the effectiveness of the IT security controls of the program or project.

7.2.9.3. Upgrades During the Operational Phase

7.2.9.3.1. A system upgrade may result in a significant change to the system configuration and security controls. Examples of significant changes include, but are not limited to, relocation to other facilities, major modification of the existing facilities, introduction of new equipment, addition or deletion of external interfaces, changes to system network connectivity, installation of new operating system software, patches to applications, new releases of software, installation of new application software, introduction of more sensitive data, or a substantial change to the system's risk posture that might affect others on the same network.

7.2.9.3.2 If a significant change occurs before the next scheduled 3-year review, the line manager must ensure that knowledgeable officials conduct a review of the security controls and re-authorize processing. Results from the review should be documented and attached to the IT Security Plan.

7.2.10. Retirement and Rollover (Disposal of Assets)

7.2.10.1. Once the system or application becomes obsolete and is to be disposed of, the line manager should notify the NARA ISO. (See Records Retention Schedules, for retention requirements and procedures for specific data and information that needs to be retained.) Current Federal regulations permit giving surplus IT equipment to organizations outside the NARA community. Other organizations are not authorized to possess copyrighted software licensed to NARA or restricted information (e.g., information whose distribution is controlled) that may be stored on NARA's surplus media.

7.1.10.2. Releasing copyrighted software or restricted information outside of NARA, even accidentally, may expose NARA Management and the Government to considerable liability. Before disposing of any storage media, NHT must do the following:

Ensure that all software and information with release restrictions are erased by overwriting the media at least once. If you have any questions concerning releasing information, data, or software on obsolete storage media, erase the media before releasing it for disposal.

Ensure that media which do not permit overwriting is destroyed or transferred to another authorized user within NARA.

Deleting information and erasing information are very different. Deleting usually removes only pointers to the files that contain the information. The information remains on the medium and may often be recovered using widely available software tools. Erasing overwrites the data fields so that the information is truly gone.

8. Records Management in the Life-Cycle Process

8.1. Purpose

Electronic systems create records that must be managed. These records may be data accumulated, processed and accessed by a system; they may also be data generated and stored by a system while it fulfills its purpose. The most effective way to manage these electronic records is to build records management requirements into systems during the design and development phases. This section provides detailed guidance on how to integrate records management awareness and compliance into NARA's Capital Planning Process and the System Development Lifecycle. A high level overview of this is depicted in Figure 8.1 below.

The procedures outlined here are organized into 6 sections corresponding to 6 key areas or choke points where Records Management review and approval can be established within the system development process. This review will ensure that sound records management practices are being incorporated into the system development lifecycle of any proposed IT system.

Each section includes questions that the Product Owner should be considering and addressing during the corresponding development phase, any identifiable products or evidence that NH can use to ensure that the system development complies with sound records management practices, a listing of results that the both the Product Owner and the NARA Records Officer should expect as a result of this phase of development, and some guidance for the Product Owner to follow during the phase. The guidance includes what is expected of the product owner and also outlines what interactions and approvals should occur during the phase.

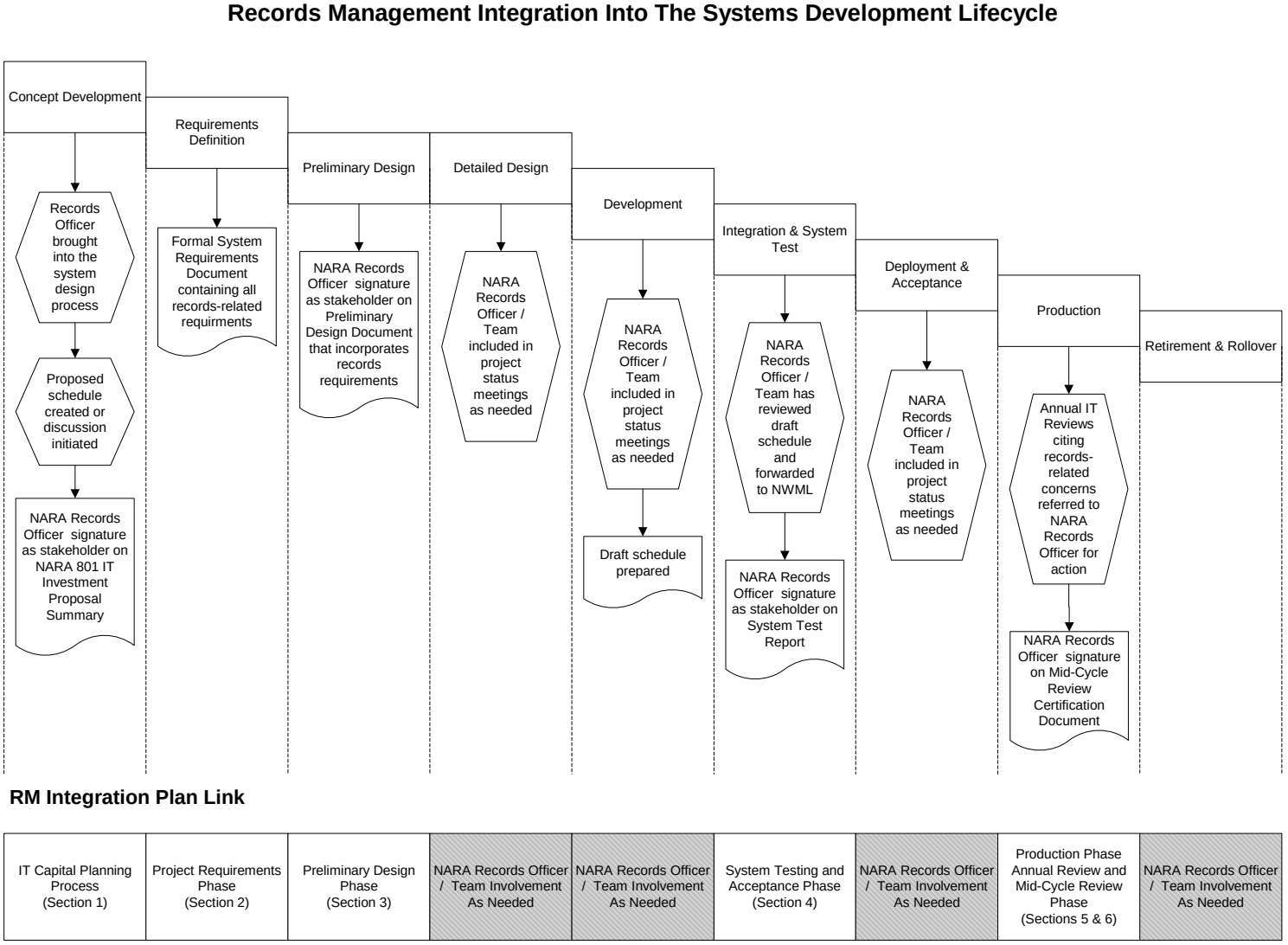


Figure 8-1. Records Management Integration Into The Systems Development Lifecycle

8.2. The IT Investment Proposal Phase

This phase is covered in Directive 801 but is also referenced here for completeness.

Questions for the Product Owner and/or Project Manager to ask:

- Is the NARA Records Officer already involved in the process? If not, contact the Records Officer immediately.
- Is this system replacing a paper-based records system? If yes, are the paper records already scheduled? If yes, can the existing dispositions be used as the basis for new dispositions for this system?
- Is this new system replacing an existing electronic system? If yes, is this existing system already scheduled? If yes, can the existing dispositions be used as the basis for new dispositions for this system?
- If the business process/workflow in question has or will be redesigned prior to system development, will these new business process methodologies account for a change in the nature of existing records? If yes, contact the Records Officer immediately.
- Is the product owner familiar with the records management requirements in the SDLC & aware of the need to incorporate them in the system design? If not, consult the NARA Records Officer.

Evidence of compliance:

- NARA Records Officer signature as stakeholder on the NARA 801 IT Investment Proposal Summary.

Results:

- Records Officer is directly involved in the system design process.
- Product owner is aware of the need for integrating records management into the system.
- A proposed records schedule has been created or a discussion concerning scheduling records has been initiated.

Guidance:

The Product Owner should consult with the NARA Records Officer to begin the process of determining the disposition(s) that will apply to the records created or maintained by the system, and then seek guidance on the requirements that should be built into an information technology

system. A number of these requirements may be the direct result of the implementation of specific dispositions.

8.3. The Project Requirements Phase

Identification and definition of project requirements occurs within the SDLC Project Initiation Phase.

Questions for the Product Owner and/or Project Manager to ask:

- Have you identified all records-related requirements that must be incorporated into the system design phase?

- Have you identified or developed additional records requirements based on your business needs?

- Have you considered how the proposed records dispositions and retention times might impact the system's records-related requirements? And, vice versa, have you considered how records-related requirements may affect proposed dispositions and retention times, especially in the area of long-term storage?

- Have you made sure that all of the records-related requirements detailed in your system requirements document are incorporated into the records requirements section of your draft IT Investment Proposal?

Evidence of compliance:

- A formal system requirements document that includes all records-related requirements.

Results:

- All records-related requirements have been identified and formally documented in the system requirements document and are fully represented in the draft IT Investment Proposal.

Guidance:

In this phase, the Product Owner must fulfill two records management-related responsibilities: 1) determining the value of the records or data to be created and/or maintained by the system under development (archival vs. disposable); and 2) determining the records management requirements, both required and business driven, that must be built into the system.

The Product Owner should be aware that establishing and implementing records retention instructions will result in a set of requirements that must be included in the system design.

The Product Owner should consider the possible uses and value of the material to the creator, including administrative, fiscal, and legal values. Consideration should also be given to whether or not the material has historical value. This information will be crucial in determining the proposed disposition of the records produced by the system and how they will need to be stored.

Using the information gathered about the records and the processes they will document and support, the Product Owner will take a first cut at determining the value of the records and establish proposed retention periods. The Product Owner will then translate these proposed retention periods into records management requirements. This process may include any of the following activities, depending upon the complexity of the system and the proposed schedule: review of the proposed records that will be created or maintained by the system; discussions with the departments program, technical, and administrative staff to gather information on the records themselves; discussions with NARA records management staff to learn about records management requirements; review of related accessioned records; review of previously scheduled records; and review of technical requirements and issues. Additionally, any NARA units having a "stake" in the records to be created or maintained by the system should be involved in the development and review of any records management-related requirements that might affect them.

The Product Owner should be aware of any NARA policy and guidance regarding the creation, use, maintenance, disposition, custody, processing, storage, reproduction, access to, or any other aspect of the lifecycle of the records that will be produced by the proposed system. The Product Owner should also determine any records management requirements generated by business need.

In order to determine records requirements based on business needs, the Product Owner will need to analyze the information about internal program need for the records, external recordkeeping requirements, and any existing records schedules.

All records management requirements will be captured in the Concept of Operations document for the system. The Concept of Operations describes the proposed system in terms of the user needs it will fulfill, its relationship to existing systems or procedures, and the ways it will be used.

It will be the Product Owner’s responsibility to ensure that information technology systems are constructed, maintained, and updated in ways that support the records management and archival requirements for the records that are handled by the systems. The NARA Records Officer will provide guidance and eventually certify that the system meets all necessary records management requirements.

8.4. The Preliminary Design Phase

Design activities occur within the SDLC Development Phase.

Questions for the Product Owner and/or Project Manager to ask:

- Has the draft schedule for the system been finalized and circulated for internal clearances?

- Has the NARA Records Officer/NARA Records Management Team been involved in schedule creation process? If not, they should review your draft schedule before continuing with development to ensure that all of the identified records-related requirements are covered by the schedule.

- Have all of the identified records management requirements been integrated into the system design?

Evidence of compliance:

- NARA Records Officer signature as stakeholder on the preliminary system design document.

Results:

- Records requirements outlined in the system requirements document have been incorporated into the preliminary system design document.

- Creation of a draft records schedule for system has been initiated.

Guidance:

During this phase of development, the Product Owner should draft a schedule for the proposed system based on the proposed schedule and findings of the requirements gathering activities. They should work closely with the NARA Records Officer to ensure that all records-related elements are covered by the draft schedule.

Once the draft schedule has been finalized it is the NARA Record Officer's responsibility to obtain all necessary agency internal clearances when appropriate. This process includes circulating a draft schedule through any appropriate offices and personnel within NARA in order to obtain their comments and concurrence. Once any necessary internal clearances have been obtained, the NARA Records Officer will submit the draft schedule to NWML for review and approval by the Archivist.

8.5. The System Testing and Acceptance Phase

System Testing and Acceptance is conducted within the SDLC Development Phase.

Questions for the Product Owner and/or Project Manager to ask:

- Has the draft schedule been reviewed by the NARA Records Officer and submitted to NWML for the review and approval by the Archivist?

- Does the system meet all of the identified records-related requirements outlined in the requirements document? If not, did you consult with the NARA Records Officer to determine the appropriate course of action?

- Has the Configuration Control Board (CCB) process taken records management issues into account for all changes?

Evidence of compliance:

- NARA Records Officer signature as stakeholder on the system test report.

- NARA Records Officer or Team has reviewed and forwarded the draft schedule to NWML for approval by the Archivist.

Results:

- System meets both the general and system-specific records management requirements identified during the requirements analysis activity.

- System is scheduled or a draft schedule has been submitted to NWML for review and approval by the Archivist.

Guidance:

During this phase of development the NARA Records Officer will formally review the project testing documentation (test plans, results, and scripts) and the draft schedule to ensure that the system meets or exceeds all identified general and system-specific records management requirements identified during the requirements analysis activity.

It is the Product Owner's responsibility to ensure that the system has been scheduled or that the draft schedule for the system has at least been submitted for approval.

If the NARA Records Officer is satisfied that all necessary records-related requirements have been met and a schedule has been approved or submitted for approval, the system will be formally approved for deployment (Go Live). Without the approval of the NARA Records

Officer, the system will not be allowed to be deployed until all records-related deficiencies are corrected.

If approval for deployment has been granted while the schedule is still in the approval cycle, modifications to existing records requirements or the addition of new requirements may be required. If so, the NARA Records Officer will work with the Product Owner and NWML to ensure that any required changes or additions are documented and addressed.

8.6. Production Phase Annual Review

The production phase annual review tasks are conducted during the SDLC Operation Phase. The Production Phase Annual Review is primarily covered in Directive 801, but is also referenced here for completeness.

Questions for the Product Owner and/or Project Manager to ask:

- Is the system content consistent with what was intended (e.g., is the system maintaining more or less data than intended? Different data?)? If not, the records schedule may need to be modified. Contact the NARA Records Officer for assistance.

- Has a work process provided or supported by the system been changed? Does this change have any records-related implications (e.g., changes to the records schedule, revised records management system requirements)? If so, contact the NARA Records Officer for assistance.

Evidence of compliance:

- [Answers to above questions listed on the annual security questionnaire submitted to IT Security & forwarded to RMT]

Results:

- Schedule has been validated or modified to reflect the findings of the assessment.

- Records management requirements have been validated or modified and then recertified by the NARA Records Officer.

Guidance:

Once a system goes live, incremental changes may occur as users build experience with it. Some of these changes can result in changes to the nature of the data being created and retained; some data fields may be abandoned, others may be added. This may require revising the data's retention period(s) or even changing the value of the data (permanent vs. disposable). The

changes can also require revisions or additions to system records management requirements. Identifying these changes early in the production phase makes it easier to respond to. Therefore, the Product Owner will report any such modifications to the system **annually** through a records management section included in the annual security review questionnaire. The NARA Records Officer will then respond appropriately, requiring either revisions to the approved records dispositions or revisions or additions to system records management requirements. This annual review is not meant to be comprehensive. Rather, it is meant as a first and early response to changes. A full, free-standing review is conducted in the next phase, the Mid-Cycle Review.

8.7. The Mid-Cycle Review Phase

Mid-Cycle review tasks occur within the SDLC Operation Phase. The Mid-Cycle Review is addressed in Directive 801, but is referenced here for completeness.

Questions for the Product Owner and/or Project Manager to ask:

- Are records management requirements being followed?
- Has the system been modified or is it being used to support different business needs than originally intended? If so, a new product plan and/or schedule may be required.
- Is the system content consistent with what was intended to be? If not the records schedule may need to be modified?
- Do the records produced by the system have the same value that was originally assigned to them? If not, the schedule may need to be modified, including retention times.

Evidence of compliance:

- NARA Records Officer signature as recertifying officer on the system's production performance review.

Results:

- Schedule has been validated or modified to reflect the findings of the assessment.
- Records management requirements have been validated or modified and then recertified by the NARA Records Officer.

Guidance:

The intent of this phase, which will occur roughly 3 years after initial system deployment (Go Live), is to ensure that the system is operating as designed and to identify any necessary changes to the records requirements or schedule resulting from any modifications made to the system after initial deployment. The NARA Records Officer will gather information about the adequacy,

effectiveness, and efficiency from the Product Owner in order to determine the impact of any modifications or upgrades made or being considered for the system and how they will affect the records-related requirements or schedule.

Based on the findings from the information gathering activity outlined above, the NARA Records Officer will either recertify the system as being in compliance or notify the Product Owner of any deficiencies or required schedule changes. If the changes are significant enough, the Product Owner may be required to submit a new product plan for the modified system as well as a revised schedule. This new product plan will need to address any new or changed records-related requirements and how the system will address them.

8.8. System Retirement/Shutdown Phase

System Retirement/Shutdown is grouped with the SDLC Operation Phase. System Retirement/Shutdown is addressed in Directive 801 but is also referenced here for completeness.

Questions for the Product Owner and/or Project Manager to ask:

- Is the system being totally shut down or will some of the data be migrated to a successor system?
- If some of the data are being migrated, will functionality be maintained and complete access to the data not migrated be continued?
- If the system will be totally shut down, will the data be maintained and kept accessible for the full retention period?
- Is the current disposition(s) for the system still appropriate? Have business needs sufficiently changed to warrant a reexamination of both the value of the records and the retention period?

Evidence of compliance:

- NARA Records Officer's signature on the shut-down documentation certifying that all records management concerns have been addressed.

Results:

- The system owner is accountable for the records contained in the system being shut down and that accountability is fully documented.
- Records are retained and kept accessible for the full retention period.

Guidance:

When an information system is shut down, the system owner must account for the records the system contains. If there is no successor system and no data are being migrated, then all data must be managed in accordance with the appropriate disposition. Data must be stored in formats and on media that will allow for complete access and usability for the full retention period specified by the records disposition. If the data are being migrated to a successor system, the owner must assure that the new system will make the data totally available and usable through the new system. If only a fraction of the data is being migrated, the remaining data still need to be maintained and kept accessible in accordance with the approved disposition.

Before a system is shut down, the NARA Records Officer must certify that the above requirements have been met and that the data from the defunct system is being properly managed and maintained.

9. PROJECT DATA DELIVERABLES

NARA application projects create several key data analysis and design documents during the Systems Development Life Cycle (SDLC). This section describes the content and format of those data documents. For further information on procedures for constructing and reviewing these documents, refer to the NARA IT Solutions Development Handbook.

9.1. Data Business Rule

Definition	A statement about a characteristic of a business entity. A business rule may be a definition, an attribute, or a description of the relationship of a business entity with itself or with another business entity. It is a constraint that governs the way business is conducted in functional areas.
Purpose	• Identifies and defines the scope and nature of the data to be developed in the application. • • Identifies potential impact on the NARA Data Model (the enterprise model). • • Further establishes a basis for communication and understanding of the scope of data in the proposed application.
Deliverables	• A textual list, definitions and attributes of the business entities and relationships to be created, read, updated, or deleted by the application.
Responsibility	• Project team, contractor, or software vendor
SDLC Development Activity	• Requirements Development Activity
SDLC Review Activity	• Requirements Review
Guidance	*The following criteria apply to the data business rules:* • The data business rules should make sense to non-technical business users. • Relationships among the business entities should be described using terminology familiar to the users. • Business entities and business relationships should be described in enough detail as to be unambiguous. • Business entities should be singular in nature. •

9.2. Conceptual Data Model (CDM)

Definition	A high-level description, understandable by non-technical NARA management and staff, of the business entities about which data will be used or created by the proposed application, and the significant business relationships among those business entities that the application will support.
Purpose	• Identifies and defines the scope and nature of the data to be developed in the application. • Identifies potential impact on existing databases and the NARA Data Model (the enterprise model). • Establishes a basis for communication and understanding of the scope of data in the proposed application.
Deliverables	• A textual list and definitions of the business entities and relationships to be created, read, updated, or deleted by the application. • A high-level business entity/business relationship diagram.
Responsibility	• Project team, contractor, or software vendor
SDLC Development Activity	• Concept Development Activity
SDLC Review Activity	• Concept Review
Guidance	*The following criteria apply to the Conceptual Data Model (CDM):* • The CDM should make sense to non-technical business users. • At a minimum, the CDM must list business entities and their definitions. A high level entity relationship diagram is strongly recommended, though not mandatory. • Business relationships among the entities should be described following business terminology familiar to the users. • Business entities and business relationships should be

described in enough detail as to be unambiguous.

- Business entities should be singular in nature.
- Many-to-many relationships are acceptable.

9.3. Logical Data View (LDV) (Optional, use if needed)

Definition	Logical entity-attribute-relationship data models representing data to be created, updated, read, or deleted as part of an application system, as seen from the perspective of a particular user, group of users, data store, or business transaction. Logical Data Views are synthesized into the Logical Data Model (see "Logical Data Model").
Purpose	• Provides a simple, consistent approach to collecting detailed user data requirements for consolidation into the Logical Data Model.
Deliverables	• Textual documentation of the data entities, attributes and relationships to be created, read, updated, or deleted by the application, as seen from a particular user or transaction perspective. • Logical entity-attribute-relationship diagram(s) in First Normal Form.
Responsibility	• Project team or contractor
SDLC Development Activity	• Requirements Definition Activity
SDLC Review Activity	• Requirements Review
Guidance	*The following criteria apply to Logical Data Views:* • Existing data entities, attributes, and relationships from the NARA Enterprise Data Model (NDM) should be reused as much as possible in constructing Logical Data Views. • New data entities, attributes, and relationships must follow NARA data naming standards.

9.4. Logical Data Model (LDM)

Definition

A third normal form representation of logical data entities, attributes, and relationships that represents the inherent structure of that data and is independent of both individual applications of the data and the software or hardware employed in representing and using the data. If LDVs have been created, the LDM is a composite of the views transformed into third normal form.

Purpose

Provides a foundation for the design of sharable physical databases.

Deliverables

- Textual documentation of the data entities, attributes and relationships to be created, read, updated, or deleted by the application.
- A Third Normal Form (3NF) logical entity-attribute-relationship diagram.

Responsibility

- Project team, contractor, or vendor

SDLC Development Activity

- Preliminary Design Activity

SDLC Review Activity

- Preliminary Design Review

Guidance

The following criteria apply to the Logical Data Model (LDM):

- The LDM must support application business rules.
- The LDM should make sense technically to the Data Administrator and to NARA application developers.
- Definitions and valid values must conform to NARA data standards.
- Entity, attribute, and relationship names must conform to NARA data naming standards.
- Repeating groups of attributes (such as a list of skills for an employee) should be broken out into separate (attributive) entities.
- Entity identifiers (one or more attributes) should have values that are unique for each entity occurrence, and are never null.
- Each attribute should be assigned to only the entity that it describes.
- All many-to-many relationships must be resolved.
- All known attributes should have been identified for

each entity.

9.5. Data Model Implementation Strategy

Definition

A textual report that describes physical changes required for performance reasons in a way that highlights any potential negative impact on business rules and data standards.

Purpose

- Factors processing constraints such as response time, cost, capacity, location, etc. into the logical data design.
- Highlights modifications needed to the Logical Data Model to meet business-processing requirements.
- Balances tradeoffs and determines acceptable physical compromises.

Justification

Logical data models are frequently modified during physical design to improve performance. Techniques used to improve performance include:

- Table splitting
- Table combining
- Data replication
- View materialization (converting SQL logical join or project views into persistent physical tables)
- Design restructuring (moving rule enforcement from the data base to the software or vice-versa, or implementing 1:m relationships via tables rather than keys)

Several of these techniques can result in denormalization, and denormalization undermines the objective of data modeling. Denormalization is the violation of first, second, and/or third normal forms resulting from logical data modeling. Denormalization creates processing anomalies which may:

- Prevent data from being created unless other data has first been created
- Allow undesired data to be retrieved
- Prevent updates from working as wanted
- Allow deletes to inadvertently remove data

The Data Model Implementation Strategy report makes visible the potential negative impact of denormalization, allowing a close scrutiny of the ramifications.

Deliverables

- Textual documentation describing reasons for each

	denormalization (e.g., logical transaction data access path mappings, volumetrics, etc.).
Responsibility	• Project Team
SDLC Development Activity	• Preliminary Design Activity
SDLC Review Activity	• Preliminary Design Review
Guidance	

9.6. Data Implementation Strategy

Definition	Description of the strategies the application will take in implementing databases.
Purpose	Describes how the application design conforms to the NARA IT Architectural policies in terms of data sharing, reuse, and security.
Deliverables	A report outlining the following strategies: • *Data Sharing* - Identification of data common to other applications. If some of the data is common but incompatible due to format or domain reasons, describe how the data will be transformed (for example, by middleware). • *Data Distribution* - If there are performance/cost reasons to distribute the data, this describes the strategy for partitioning and maintaining the integrity of the replicated data. • *Data Migration* - If there will be legacy data or data from external sources used to initially populate the database, this describes what approach will be taken to bring that data in, and especially if quality assurance of the data (cleansing, etc.) is to be done prior to the migration. • *Data Security/Sensitivity/Access* – If any of the data is classified or subject to privacy laws, deed restrictions, etc., this describes the approach being taken to ensure these requirements are met.
Responsibility	• Project team, contractor
SDLC Development Activity	• Preliminary Design Activity
SDLC Review Activity	• Preliminary Design Review
Guidance	

9.7. Physical Data Model (PDM)

Definition	A model that results from a transformation of a third normal form (3NF) Logical Data Model into a set of technologically independent requirements and constraints for the physical environment of hardware, software, and network configurations in which data will reside.
Purpose	Documents the physical design of the database for implementation.
Deliverables	• Documentation of table and column designs, domain definitions, and integrity controls. • Documentation of DBMS configuration and technical management parameters such as sizing, indices, access views, locking, tuning, backup, recovery, buffering, etc. • Source data definition language statements to be used by the DBMS software to generate the actual physical database structures, referential integrity, and triggers/edits.
Responsibility	• Project team and/or contractor, with DBA as primary reviewer.
SDLC Development Activity	• Detailed Design Activity
SDLC Review Activity	• Critical Design Review
Guidance	

10. STANDARDS

10.1. Data Naming Standards

Data entities, attributes, and relationships will be assigned two names, namely a fully spelled out business name, and a shortened name that, when converted to a physical name, will conform to the length required by the appropriate DBMS. The following guidance applies to the short names.

10.1.1. Entities

An entity is a person, place, thing or concept of significance about which facts are to be recorded. There are three basic types of entities: fundamental, associative, and attributive. Supertypes and subtypes are forms of fundamental entities. All these are described below. Standards for naming and describing all types of entities are listed here as well.

10.1.2. Entity Types

10.1.2.1. Fundamental Entity

A fundamental entity is an entity that is non-derivable, and is independent of any other entity for its existence.

Examples: ORGANIZATION, PERSON, HOLDING

10.1.2.2. Associative Entity

An associative entity is an entity that represents a relationship between two or more entities. An associative entity does not exist independently of its related entities. It further describes the relationship between the entities with attributes that are often referred to as intersecting data.

If an associative entity has an understandable business name, it should be used. If no business name is apparent, the associative entity name should be a concatenation of the parent entity names.

Examples: EMPLOYMENT, ORGANIZATION HOLDING

10.1.2.3. Attributive Entity

An attributive entity is an entity that characterizes only one other entity, often referred to as its parent. An attributive entity is formed by removing repeating fields (attributes) from its parent entity. An attributive entity can have relationships with other entities in addition to the relationship with the parent entity.

The name of an attributive entity always refers to its parent entity. The name must be the parent entity name followed by a noun that differentiates it from the parent.

Example: PERSON IDENTIFIER, ORGANIZATION NAME

10.1.2.4. Supertype Entity

A supertype entity is an occurrence of a fundamental entity that can be related in a hierarchy to two or more subtype entities. All of the attributes and relationships of the supertype entity are inherited by its subtype entities.

Example: Supertype PARTY, which has attributes of name and address. It can have subtypes of PERSON and ORGANIZATION.

10.1.2.5. Subtype Entity

A subtype entity is an entity that inherits all the attributes and relationships of its supertype entity. In addition, a subtype entity has at least one unique relationship with another entity or at least one unique attribute.

Example: Subtypes PERSON and ORGANIZATION inherit the attributes of name and address, but also have unique attributes of their own, such as Birth Date for PERSON, and Organization Type for ORGANIZATION.

10.1.3. Naming Entities

- An entity's name must be a noun with or without modifiers preceding it.
- The name must be familiar and easily recognizable to business users.
- Nouns must be singular.
- Verbs are not allowed as modifiers.
- Possessive nouns must not be used in the name.
- Proper nouns may be used in the name if generally accepted.
- Allowable characters are uppercase A-Z, the space character, and numerals 0-9. The first character must be a letter. Punctuation marks or special characters, including the slash and the hyphen, are not allowed.
- Abbreviated names and acronyms commonly used at NARA are allowed. Look for an approved abbreviation or acronym on the list maintained by the Data Administrator.
- The maximum length of the entity name is determined by modeling tools. Use the complete name if it fits.

10.1.4. Describing Entities

The following apply when describing an entity of any type (fundamental, associative, attributive, supertype, or subtype):

- An entity description must be a noun phrase. If necessary, one or more sentences shall be added to complete the description.
- The description must pertain to a single entity name.
- The description must be clear, concise, and unambiguous. The description shall be broad enough that no instances of the entity are omitted.
- The description shall be relevant to its business purpose and independent of technology and implementation.
- The description shall be stable over time.
- The description shall not repeat the name of the entity at the beginning of the sentence.
- The description must be of an entity, not of the data NARA captures about the entity, nor the functions, applications, or organizations that use or create the data.
- Avoid the use of acronyms or abbreviations.
- The description must be as follows:

- Grammatically correct
- Spelled correctly
- Complete, accurate, and fully reflecting the meaning of the entity
- Written in active voice

10.1.5. Attributes

An attribute is a single item of information that describes an entity.

Examples: birthdate, age

10.1.6. Naming Attributes

- The name must be a noun with the name of the entity preceding it.
- The name must be familiar and easily recognizable to the users.
- The first character of the name must be alphabetic.
- The last word in the attribute name must be a class word indicating the type of information it represents. See the list of class words and their definitions and usage. An exception to this rule occurs when the attribute name is the same as a class word and needs no modifiers. In this case spell out the full word once and do not add an abbreviated class word at the end. For example, use code (not code cd), and name (not name name) as the attribute name.
- Nouns must be singular except where the plural form is commonly used, as in total sales amt.
- Possessive nouns must not be used in a name.
- Proper nouns may be used in the name if generally accepted.
- Allowable characters are lowercase a-z and the space character. Punctuation marks or special characters, including the slash and the hyphen are not allowed. Numerals 0-9 are also allowed.
- The maximum length of the name is determined by modeling tools.
- Do not abbreviate (except for class words) unless necessary to fit the length limitation.
- Abbreviated names and acronyms are allowed, if necessary. Look for an approved abbreviation or acronym on the abbreviation list later in this document. If the terms to be abbreviated are not in the list, follow the abbreviation procedures described with the list.
- If a sequence or revision number attribute is required to record a series of instances, use:

  revision nmbr if the new entity revises or replaces the information of its predecessor, or

  seq nmbr if the entities are a series of new data over time, but several may be active at the same time.

10.1.7. Describing Attributes

- The description must be a noun phrase or a noun clause. If necessary, one or more sentences shall be added to complete the description.
- The description must describe the attribute with respect to its entity, but must not be redundant with the entity description or attribute name.

- The description must be clear, precise, and unambiguous. It must identify the attribute and distinguish it from any other actual or possible attribute.
- An attribute shall be defined in terms of business rules, not in terms of information processing rules or physical considerations.
- Abbreviations must not be used in descriptions. Acronyms shall be used sparingly.
- Two attributes shall not be circular in description (i.e., attribute description sets cannot exist where one description points to a second description; and the second attribute description points back to the first description).
- The description must be:

  - Grammatically correct
  - Spelled correctly
  - Complete, accurate, and fully reflecting the meaning of the attribute
  - Written in active voice, where possible

- Include an example. If the object occurrence concerns data, include some sample data values.
- Assess description quality. Confirm that the description communicates its business content to both business and technical users.

10.1.8. Relationships

A relationship is an association between two entities or between two occurrences of the same entity. A relationship between two entities is called a non-recursive relationship. A recursive relationship is a relationship between two occurrences of the same entity. Relationships are used in entity relationship diagrams to convey information as to how entities correspond to one another.

10.1.9. Naming Relationships

- Each relationship must ordinarily have two names, one with an active verb and one with a passive verb.
- Each relationship name must begin with a verb in the present tense; the terms "MAY" or "MUST" shall not be included in the final form of the relationship name.
- When the entity names and the relationship names are combined in the following formats, they must form two sentences, each describing the relationship from the point of view of one of the entities:
- ENTITY-NAME-A *relationship-name-a* ENTITY-NAME-B
- ENTITY-NAME-B *relationship-name-b* ENTITY-NAME-A
- If the relationship involves only one entity (i.e., a recursive relationship), the entity name goes at the beginning and end of both sentences.

Examples of non-recursive relationships:

EMPLOYEE	*receives*	PAYMENT
PAYMENT	*is made to*	EMPLOYEE
RESIDENCE	*is owned by*	PERSON
PERSON	*owns*	RESIDENCE

Example of a recursive relationship:

EMPLOYEE *supervises* EMPLOYEE
EMPLOYEE *is supervised by* EMPLOYEE

- Relationship names shall be clear and precise.
- Allowable characters for a relationship are lowercase a-z and the space character.
- No punctuation marks or special characters, including the slash and the hyphen (-), shall be used.
- Numbers are not allowable characters.
- Verbs occurring in a relationship name shall be singular unless the sense requires the plural.
- Abbreviations and acronyms must not be used in relationship names.
- Avoid use of verbs that tell nothing about the nature of the relationship, such as "is" or "has". For associative entities that take on a variety of different roles, it is all right to use "involves" as the relationship verb.

10.1.10. Describing Relationships

Identify the nature of the relationship between the entities. For example, the entities EMPLOYEE and WORKSITE may have the following relationship:

EMPLOYEE *works at* WORKSITE

WORKSITE *is work location of* EMPLOYEE

- Identify the optionality between the entities. Relationship optionality indicates whether a relationship is optional or required. Frequently, a relationship can be optional when viewed from one entity and required when viewed from the other.
- Identify the cardinality between the entities. Relationship cardinality indicates how many of one entity is related to how many of another entity. Relationships between two entities may be one-to-one (1: 1), one-to-many (1:M) or many-to-many (M:M).

10.1.11. Class Words

Class words are special terms added at the end of every attribute short name to indicate the type of information that the attribute represents. This class word will uniquely categorize an attribute. The definition of the class word must be unambiguous. This will ensure that a class word will not overlap with another class word. The abbreviated form of the class word must be used as the last word in an attribute short name, unless there are no other modifier words needed. When the attribute short name consists solely of the class word, use the full word, not the abbreviation.

The Data Administrator maintains the following list of approved class words. Only the Data Administrator may add new class words. If an additional class word is needed, submit the proposed class word and its abbreviation to the Data Administrator for inclusion in the list of approved class words.

Class Word	Abbreviation	Definition
Abbreviation	ABBREV	A shorter version of a word.
Amount	AMT	A value used for floating point real numbers (non--integer) and currency.
Code	CD	A combination of one or more numbers, letters, or special characters which represents a specific meaning. A CODE represents a finite, predetermined value or status that may apply to many objects.
Count	CNT	A number of occurrences or summation of objects which may be represented by an integer.
Date	DT	A specific day represented by calendar conventions. Includes month, day and year. Use MONTH YEAR DT for a date that indicates just the month and year but not the day.
Datetime	DATETIME	Date and time stamp for historical tracking or to differentiate instances of attributive or associative entities.
Day	DY	A day of the month.
Description	DESC	A single line of free-form text used to define, comment upon, or describe the object. (See also TEXT.)
Dimension	DIMNSN	A measured linear distance that is one dimensional.
Flag	FLG	An attribute whose values can only be yes or no.
Identifier	ID	A combination of letters, numbers, and/or special characters used to uniquely designate each occurrence of one entity.
Month	MNTH	A month of the year.
Name	NAME	One or more words used to identify a particular person, place or thing. This class term must be used when there is a name lookup table associated with the attribute.
Number	NMBR	A combination of letters, numbers, and special characters used to designate one person or thing. A NMBR does not have to be unique, as opposed to an ID, which is always unique. It is non-quantitative, and therefore cannot be used in calculations.
Percentage	PCT	Number of parts in or to every hundred.
Quantity	QTY	A non-monetary numeric value that does not have to be a whole number. It is a calculated or aggregated value.
Quarter	QTR	A 3-month time period beginning with the first, fourth, seventh, or tenth month of a calendar or fiscal year.
Rate	RATE	A quantity, amount or degree of one value in relation

		to units of another type of value. Example: Miles per gallon RATE, Interest RATE.
Text	TXT	Multiple line free-form text used for comments, definition or describing an object. (See also DESCR.)
Time	TIME	A specific chronological point using clock conventions. (Use AMT for time duration)
Title	TITLE	An official designation.
Type Code	TYPE CD	A combination of one or more numbers, letters or special characters which represents a specific meaning. Each value of a TYPE CODE identifies a specific kind of instance of the entity.
Year	YR	A calendar of fiscal year.

10.1.12. Acronyms and Abbreviations

An *acronym* is a word formed from the first (or first few) letters of each word of a series of words. An *abbreviation* is a shortened form of a word or phrase by contraction or by omission of letters.

- The abbreviated name shall have each word in the name abbreviated in accordance with this section.
- The business name shall never be abbreviated.

10.1.13. Candidate Abbreviations

Create a candidate abbreviation by using the following rules. Submit the full name and the proposed abbreviation to the Data Administrator for approval and inclusion in the list of official abbreviations.

- Check the following sources for a common abbreviation for the term in question:

  - Any commonly accepted abbreviation (de facto standard)
  - The lists of NARA abbreviations and class words.

- An abbreviation for the term in question may be found through these sources, or follow abbreviation rules below. If a readily acceptable abbreviation is found, use it. If the abbreviation is not readily acceptable due to possible conflicts or duplications, or it does not adequately represent the word it replaces, continue with the abbreviation rules.
- Ensure that each abbreviation is unique, not only with regard to other abbreviations, but also with respect to acronyms.
- Ensure that each term has only one abbreviation.
- Abbreviations may consist of alphabetic characters only.
- Only the singular form of the noun or noun phrase should be used.
- An abbreviation should be recognizable; that is, looking at the abbreviation, one should be able to visualize the word.
- Preserve the first letter of the term, whether it is a vowel or consonant.
- Always treat "y" as a consonant.

- Delete unnecessary vowels; however, not all vowels need to be eliminated to have a valid abbreviation. Keep those that are necessary to make the abbreviation understandable.
- Delete one consonant of a double consonant.
- If the removal of a vowel causes a double consonant then keep the vowel.
- If the term has a leading double vowel, such as "au" or "ou," keep both vowels. For example, AUTHORIZATION would be abbreviated AUTHZN.
- If the term has trailing silent vowels, drop them; if the trailing vowels are not silent, retain the vowel sound. For example, a standard abbreviation for MORGUE would be MRG or MORG; whereas, the abbreviation for DISTRIBUTEE might be DSTRBTE.
- If the abbreviation exceeds the necessary length, delete the last of any repeated consonants (not just double consonants), and check again for duplicate abbreviations. If the abbreviation is still too long, submit a request to Data Administration to accept a deviation from the abbreviation rules. For example, to abbreviate TRANSPORTATION the first candidate abbreviation is TRNSPRTATN; a second pass may be TRNSPRTN. But a waiver must be requested if further abbreviating is needed.
- Avoid creation of abbreviations that produce words; i.e., ALTERATION abbreviated to ALTER. There may be exceptions, including "word-like" abbreviations with an internal or external history of common usage. Such exceptions must include a justification memorandum when submitting the candidate abbreviation.
- If the abbreviation already exists for another word, i.e., FCLTY for FACILITY, then it is necessary to either keep one of the vowels for the new abbreviation, or use a commonly accepted abbreviation that is sufficiently different. For example, abbreviating both FACILITY and FACULTY would lead to FCLTY. An acceptable abbreviation for FACULTY might be FCULTY.
- A root word and its derivatives should have the same 'root' abbreviation. For example, the abbreviation for EXEMPT is EXMPT, and for EXEMPTION is EXMPTN. The 'root abbreviation' in both cases is EXMPT. An exception to this rule may occur when the derivative word has a prefix. For example, the abbreviation for EMPLOYMENT is EMPLYMT. The abbreviation for UNEMPLOYMENT is UNEMPLYMT. If this is too long, UNMPLYMT is still recognizable, which is one of the major considerations for an abbreviation.
- Always eliminate the vowels in a suffix. Use G as the abbreviation for the ING suffix, and MT for the MENT suffix. The abbreviation for PRINTING is PRINTG, and an acceptable abbreviation for EMPLOYMENT is EMPLYMT.
- If a root word is too short to abbreviate, its derivatives may have the root portion spelled out or abbreviated, but, if abbreviated, all derivatives must have the same abbreviation of the root portion of the word. For example, if PRINT is not abbreviated, then PRINTING would be abbreviated PRINTG, and all other derivatives would also contain the root PRINT. On the other hand, if CLEAR is not abbreviated, and CLEARANCE is abbreviated CLRNC, then CLR must be used as the root for all derivatives of CLEAR.
- Abbreviations must not spell an expletive.
- If necessary, join two words (for example, ITEM TYPE would become ITMTYP).

10.1.14. Standard Logical Data Modeling Abbreviations

Full word/phrase	Normal abbreviation
Account	ACCT
Activity	ACTVTY
Actual	ACTL
Address	ADDR
Affiliation	AFILTN
Agreement	AGREMT
Authority	AUTHRTY
Business	BUSNS
Calendar	CALNDR
Calendar Period	CALPRD
Category	CATEGY
Classification	CLASFCTN
Communication	COMUNCTN
Completion	CMPLTN
Conversion	CNVRSN
Detail	DTL
Distribution	DISTRBTN
Document	DOCMT
Effective	EFCTV
Electronic	ELCTRNC
Employee	EMP
Employment	EMPLMT
Estimated	ESTMTD
Expected	EXPCTD
Facility	FCLTY
Frequency	FREQ
Function	FNCTN
Group	GRP
Hours	HRS
Item	ITM
Language	LANG
Location	LOCTN
Locator	LOCTR
Maintenance	MAINTNC
Material	MATRL
Milestone	MILSTN
Notification	NOTIFCN
Occurrence	OCCRNC
Organization	ORG

Person	PERS
Position	POSTN
Probability	PRBLTY
Problem	PRBLM
Product	PRDCT
Property	PROPTY
Recommend, Recommended	RECMND
Referred	REFRD
Remark	RMRK
Response	RSPNS
Review	REVW
Sequence	SEQ
Service	SRVC
Significant	SGNFCNT
Special	SPCL
Standard	STNDRD
Structure	STRCTR
Supplemental	SPLMNL

10.1.15. Lookup Tables (Authority Files)

A lookup table (or authority file) is used for data validation or to look up attribute values. It is either a decode table or a name table. A decode table contains valid values for an attribute. It contains at least two attributes, an attribute for the code (which may be CODE), and an attribute that defines, describes or names the attribute it is decoding (DESC). A name table is generally a selection table and contains only one attribute, a name or value attribute.

10.2. Required Meta-Data

Since Logical Data Models must be integrated with the NARA Data Model (NDM), each application data model object should map to a corresponding NDM object. The following table describes which data model object properties are needed to perform the integration exercise. Application projects will probably document other properties for application-specific reasons.

Table contents are:

- **Data Model Object Type** (column 1) names the type of object (entity, attribute, or relationship).
- **Properties** (column 2) describes the different types of characteristics of the data model object.
- **Comments** (column 3) provides notes and descriptive guidance related to the data model object property.
- **Required** for the Conceptual Data Model (CDM), the Logical Data Views (LDV), the Logical Data Model (LDM) or and Physical Data Model (PDM) (columns 4-8) indicates if the data model object property is required in a particular category of data model.
- **NDM Match Criteria** (column 9) describes the degree to which the data model property must align with an equivalent data model object in the NARA Data Model to be considered a match with NARA Data Standards.
- **NDM Mapping Criteria** (column 10) describes the degree to which the data model property must align with an equivalent data model object in the NARA Data Model to be considered a variant of a NARA Data Model object.

Data Model Object Type	Properties	Comments	CDM	LDV	LDM	PDM	NDM Match Criteria	NDM Mapping Criteria
ENTITY								
	Business name		Required	Required	Required		Similar concept	Differs
	Short name	See "Data Naming Standards"			Required if needed		Identical	Differs
	Physical (table) name					Required		
	Type (fundamental, associative, attributive)	See "Data Naming Standards"			Required		Identical	Differs
	Definition	See "Data Naming Standards"	Required	Required	Required		Similar concept	Differs
	Primary key type	Describes whether primary key is intelligent or non-intelligent (surrogate key)			Required	Required	Identical	Differs
ATTRIBUTE								
	Business name			Required	Required		Similar concept	Differs
	Short name	See "Data Naming Standards"			Required if needed		Identical	Differs
	Physical (column) name					Required		
	Definition	See "Data Naming Standards"		Required	Required		Similar concept	Differs
	Class word	See "Data Naming Standards"			Required		Identical	Differs
	Name of entity where attribute resides	Separate item for this not needed if entity name is prime word component of attribute name (see "Data Naming Standards"			Required		Identical	Differs
	Logical length	From user requirements			Required		Same length or shorter	Differs
	Data type	Includes physical length if not implied by datatype			Required	Required	Identical	Differs
	Valid values (authority file	In the PDM, this is the			Required	Required		Identical or

Data Model Object Type	Properties	Comments	CDM	LDV	LDM	PDM	NDM Match Criteria	NDM Mapping Criteria
	name, value list, or value formation rules)	authority file physical name					Identical or subset	subset
	Security level	Contact NARA Security Architect for details			Required if needed		N/A	N/A
RELATIONSHIP								
	Business name		Required	Required				
	Short name	See "Data Naming Standards"			Required if needed		Similar concept	Differs
	Definition	See "Data Naming Standards"			Required		Similar concept	Differs
	Entities involved in relationship				Required		Identical	Differs
	Cardinality				Required		Identical	Differs
	Inter-relationship constraints	Textual description of complex relationships between this relationship and other relationships (e.g., rules such as 'exclusive OR' that cannot be conveniently expressed graphically)		Required	Required if needed		Similar concept	Differs
	Primary and foreign keys				Required	Required	N/A	N/A

10.3. Examples

10.3.1. Data Dictionary

10.3.1.1. Data Dictionary for the Conceptual Data Model

ENTITY NAME	Definition
ORGANIZATION	An administrative or functional structure that conducts business (e.g., government agency, company, educational institution, association, church, nonprofit institution, or any organizational level thereof).
PERSON	A human being.
HOLDING	Recorded information, regardless of media or characteristics, made or received and maintained by an ORGANIZATION or PERSON in the transaction of business or the conduct of affairs and kept as evidence of such activity.

RELATIONSHIPS
An ORGANIZATION employs many PERSONs.
A PERSON is employed by many ORGANIZATIONs.
An ORGANIZATION routes many HOLDINGs.
A HOLDING is routed by many ORGANIZATIONs.

10.3.1.2. Data Dictionary for the Logical Data Model

ORGANIZATION

ENTITY NAME	ORGANIZATION
ENTITY SHORT NAME	ORG
ENTITY DEFINITION	An administrative or functional structure that conducts business (e.g., government agency, company, educational institution, association, church, nonprofit institution, or any organizational level thereof).
ENTITY TYPE	Fundamental
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	code
ATTRIBUTE DEFINITION	Identifier of an ORGANIZATION, e.g., NH for Office of Human Resources and Information Services.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	6

RELATIONSHIPS
An ORGANIZATION uses 1 or many ORGANIZATION NAMEs.
An ORGANIZATION employs 0, 1, or many PERSONs.
An ORGANIZTION routes 0, 1, or many HOLDINGs.
An ORGANIZATION includes 0, 1, or many ORGANIZATIONs.
An ORGANIZATION is included in 0, 1, or many ORGANIZATIONs.

PERSON

ENTITY NAME	PERSON
ENTITY DEFINITION	A human being.
ENTITY TYPE	Fundamental
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	first name
ATTRIBUTE DEFINITION	A name that stands first in one's full name.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

ATTRIBUTE NAME	middle name
ATTRIBUTE DEFINITION	A name that occurs between a PERSON's first name and surname.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

ATTRIBUTE NAME	last name
ATTRIBUTE DEFINITION	A name borne in common by members of a family; the surname.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

RELATIONSHIPS
A PERSON is employed by 1 or many ORGANIZATIONs.

HOLDING

ENTITY NAME	HOLDING
ENTITY DEFINITION	Recorded information, regardless of media or characteristics, made or received and maintained by an ORGANIZATION or PERSON in the transaction of business or the conduct of affairs and kept as evidence of such activity.
ENTITY TYPE	Fundamental
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	title
ATTRIBUTE DEFINITION	A name assigned to a HOLDING or group of HOLDINGSs.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	250

ATTRIBUTE NAME	inclusive start date
ATTRIBUTE SHORT NAME	inclsv strt dt
ATTRIBUTE DEFINITION	A beginning date on which the HOLDING was created, maintained, or accumulated.
ATTRIBUTE DATA TYPE	Date
ATTRIBUTE LENGTH	8

ATTRIBUTE NAME	inclusive end date
ATTRIBUTE SHORT NAME	inclsv end dt
ATTRIBUTE DEFINITION	A last date on which the HOLDING was created, maintained, or accumulated.
ATTRIBUTE DATA TYPE	Date
ATTRIBUTE LENGTH	8

RELATIONSHIPS
A HOLDING is routed by 0, 1 or many ORGANIZATIONs.

ORGANIZATION NAME

ENTITY NAME	ORGANIZATION NAME
ENTITY SHORT NAME	ORG NAME
ENTITY DEFINITION	A title of an organization.
ENTITY TYPE	Attributive
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	legal name
ATTRIBUTE DEFINITION	Title of an organization used in legal documents such as contracts, memoranda of understanding and statutes.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	100

ATTRIBUTE NAME	acronym name
ATTRIBUTE DEFINITION	A word formed from the initial letters of a name, such as NHPRC for National Historical Publications and Records Commission or by combining initial letters or parts of a series of words, such as MARC for Machine-Readable Cataloging.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	20

RELATIONSHIPS
An ORGANIZATION NAME is used by 1 and only 1 ORGANIZATION.

ORGANIZATION PERSON

ENTITY NAME	ORGANIZATION PERSON
ENTITY SHORT NAME	ORG PERS
ENTITY DEFINITION	A relationship between an ORGANZATION and a PERSON.
ENTITY TYPE	Associative
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	role name
ATTRIBUTE DEFINITION	The nature of the relationship between an ORGANZATION and a PERSON, e.g., employs.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

ORGANIZATION HOLDING

ENTITY NAME	ORGANIZATION HOLDING
ENTITY SHORT NAME	ORG HLDNG
ENTITY DEFINITION	A relationship between an ORGANIZATION and a HOLDING.
ENTITY TYPE	Associative
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	Role name
ATTRIBUTE DEFINITION	The nature of the relationship between an ORGANIZATION and a HOLDING, e.g., routes.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

ORGANIZATION STRUCTURE

ENTITY NAME	ORGANIZATION STRUCTURE
ENTITY SHORT NAME	ORG STRCTR
ENTITY DEFINITION	A relationship between two ORGANIZATIONs; represents organizational hierarchy.
ENTITY TYPE	Associative
PRIMARY KEY TYPE	Non-intelligent

ATTRIBUTE NAME	role name
ATTRIBUTE DEFINITION	The nature of a relationship between two parts of an ORGANIZATION, e.g., includes.
ATTRIBUTE DATA TYPE	Varchar
ATTRIBUTE LENGTH	50

10.3.2. Data Business Rules

The example is taken from a much larger document of the data business rules for the Order Fulfillment and Accounting System (OFAS) which is comprised of multiple subsystems. The sample refers to other parts of the document that are not included here.

10.3.2.1. ENTITY LIST for the processing of FORM 80s

ORGANIZATION: An administrative or functional structure within NARA.

CUSTOMER: An individual or organization that conducts business with NARA.

CLASS IDENTIFICATION: An arbitrary grouping of a customer.

REQUEST: A submission of a form 80 by a customer asking for a specific NARA product.

INVENTORY ITEM: A NARA cataloged inventory item available for sale. In the case of form 80s, this is almost always a microfilm to paper or paper to paper reproduction.

PAYMENT: A reimbursement for a NARA inventory item that has been purchased by a customer.

PAYMENT TYPE: A specific method of reimbursement.

INVOICE: A recording of a completed transaction.

BILL NOTIFICATION: A notice to a customer that money is owed to NARA for an inventory item(s) that was requested by the customer.

QUOTE: A request by a customer for a formal estimate of the cost of purchasing a particular NARA inventory item(s).

ORDER: A formal requisition, triggered by a request, for a specific quantity of a NARA inventory item.

BACK ORDER: A pending status of an order for a NARA inventory item that is temporarily not available in the ordered quantity.

10.3.2.2. RELATIONSHIPS for the processing of Form 80s

ORGANIZATION processes 0,1,m REQUEST
ORGANIZATION stores 0,1,m INVENTORY ITEM
ORGANIZATION receives 0,1,m PAYMENT
ORGANIZATION prepares 0,1,m INVOICE

ORGANIZATION generates 0,1,m QUOTE
ORGANIZATION generates 0,1,m ORDER
ORGANIZATION generates 0,1,m BACK ORDER
ORGANIZATION generates 0,1,m BILL NOTIFICATION

CUSTOMER submits 0,1,m REQUEST
CUSTOMER submits 0,1,m PAYMENTS
CUSTOMER receives 0,1,m INVOICE
CUSTOMER receives 0,1,m QUOTE
CUSTOMER has 1,m CLASS ID
CUSTOMER receives 0,1,m BILL NOTIFICATION

CLASS ID applies to 0,1,m CUSTOMER

REQUEST is processed by 1 ORGANIZATION
REQUEST is submitted by 1 CUSTOMER
REQUEST contains 1,m INVENTORY ITEM
REQUEST triggers 0,1,m QUOTE
REQUEST triggers 0,1,m ORDER

PAYMENT is received by 1 ORGANIZATION
PAYMENT is applied to 1,m ORDER
PAYMENT is received from 1 CUSTOMER
PAYMENT is designated by 1,m PAYMENT TYPE

PAYMENT TYPE specifies type for 0,1,m PAYMENT

INVOICE is prepared by 1 ORGANIZATION
INVOICE contains 1,m INVENTORY ITEM
INVOICE is sent to 1 CUSTOMER

BILL NOTIFICATION is prepared by 1 ORGANIZATION
BILL NOTIFICATION is received by 1 CUSTOMER
BILL NOTIFICATION contains 1,m INVENTORY ITEM

INVENTORY ITEM is stored by 1,m ORGANIZATION
INVENTORY ITEM is submitted on 0,1,m REQUEST
INVENTORY ITEM is billed on 0,1,m INVOICE
INVENTORY ITEM is listed on 0,1,m QUOTE
INVENTORY ITEM is listed on 0,1,m BACK ORDER
INVENTORY ITEM is listed on 0,1,m ORDER
INVENTORY ITEM is listed on 0,1,m BILL NOTIFICATION

QUOTE contains 1,m INVENTORY ITEM
QUOTE is triggered by 1 REQUEST
QUOTE is sent to 1 CUSTOMER

QUOTE is generated by 1 ORGANIZATION

ORDER contains 1,m INVENTORY ITEM
ORDER is paid by 1,m PAYMENT
ORDER is triggered by 1 REQUEST
ORDER is generated by 1 ORGANIZATION

BACK ORDER contains 1,m INVENTORY ITEM
BACK ORDER is generated by 1 ORGANIZATION

Note: Item to Site relationship – some items are stored only at specific sites while other items may be stored at multiple sites

10.3.2.3. ATTRIBUTE DEFINITIONS for the processing of FORM 80s

ORGANIZATION ATTRIBUTES:

Site ID – Identifies a particular NARA organization. Sample domain includes A1PUBOFF, NLC, NF, NRAB, NRPC. (See table in Form 72 processing for sample domains)

Site Description – Name of a particular NARA organization. Sample domain includes Publications Distribution Office, Jimmy Carter Library, Northeast Region – Boston, etc. (See Table in form 72 processing for sample domains)

Address – Mailing address for the organization.

CLASS IDENTIFIER ATTRIBUTES:

Class ID – An identifier that represents a general grouping of customers by a type. Class Ids include BPA, PO, DA, PUB, WALKIN, etc. Customer may have multiple classes. (See Table in form 72 processing for sample domains)

Class ID Name – The textual name of a particular class id; e.g. Public, Deposit Account, Blanket Purchase Agreement. (See Table in form 72 processing for sample domains)

Balance Type - Specifies a type of balance for a particular class id. Values are open inventory item or balance forward.

Finance Charge – Specifies whether there is no finance charge applied for a particular class id or whether the finance charge is a percentage of the balance or an amount.

Minimum Payment - Specifies whether there is no minimum payment required for a particular class id or the minimum payment is a percentage of the balance or an amount.

Maximum Write off – Specifies whether or not there is a maximum write off and maximum amount of a write off for a particular class id.

Statement Cycle – Specifies frequency of balance statements for a particular class id. Valid values are none, weekly, biweekly, monthly, bimonthly, semimonthly, quarterly. Only one can be selected.

Maintain History - Specifies the type of history to be maintained for a particular class id. Valid values are calendar year, fiscal year, transaction, or distribution. All or none may be selected.

CUSTOMER ATTRIBUTES:

Customer Identifier – Selectable for existing customers or generated through the Customer Maintenance Screen. Cannot be selected if document number and site id are blank. Customer Ids are generated through two different sources – the web interface (WOENET) and the Customer Maintenance screen in Great Plains software. The structures are very different – through the web interface, the customer id is a merging and cropping of the individual names and through the Great Plains, it is a combination of a Class ID and a sequential number. Customer may have multiple Ids.

Customer Name – Populated if document number selected or information is entered through the Customer Maintenance Screen. Fields are "Name/Company", 'Last Name", and "Statement Name". Name/Company and Statement Name are the same. Last Name is used for an individual and must contain only the last name. Name is either an individual or an organization. If an organization, then a Contact must also be specified.

Address Type – Indicates the type of address (Home, Shipping, Billing, etc.)

Address – Customer may have multiple addresses in the system. Addresses are structured in standard format (Line 1, line 2, line 3, City, State, Zip).

Phone – Customer phone number in format (000) 000-0000.

Email – Customer email address.

INVENTORY ITEM ATTRIBUTES:

Inventory Item Number – A cataloging of a NARA inventory item. Inventory items for form 80 processing are in almost all cases a form of reproduction. (Note - only sample I could find is "FORM80")

Inventory Item Description – A text field that describes the archivable inventory item being copied. For form 80 processing, this field contains the name of the archivable inventory item – e.g. NATF 80, Veterans Record; or NATF 84, Land Entry File.

Unit of Measure – The smallest unit in which the inventory item is cataloged, e.g. each.

Unit Price – Unit Price for the specific inventory item number.

REQUEST ATTRIBUTES:

Format – Format in which the request is submitted by the customer. Values include letter, phone call, email, form, etc.

Status – A status that indicates the stage of processing of the form 80. Values are: Administrative, Servicing, Production, Shipping, Mailed from mailroom, Voided, Negative Search.

Status Date – Calendar date the document was placed in the current status.

Mail Date – Date the completed package was actually mailed.

Days in Status – Number of days the form 80 has been in the current status.

Image ID – A sequential number assigned to a submitted form 80 during the scan process.

Unit/Site – (See form 72 doc)

Form ID – A 2-position indicator of the form number (e.g. 80, 86, etc).

Doc ID – An abbreviated form of the type of document for the form 80. Valid types are OFF (Order – Fixed Fee), IFF (Invoice – Fixed Fee), BOFF (Back Order – Fixed Fee), RFF (Return – Fixed Fee), ITRAN (Invoice – Transition Form 80s)

Doc Number – A number that identifies a specific form 80 transaction, comprised of the Doc ID and a 14-digit number.

Master Number – (See form 72 doc)

Item Quantity – Quantity of the item that is requested.

PAYMENT TYPE ATTRIBUTES:

Payment Type ID – Describes the format of the payment for the request. Values are CC, Check.

PAYMENT ATTRIBUTES:

Payment Type – CC, Check.

Bill Order Indicator – A yes/no flag indicating whether the customer submitted a form 80 requesting to be billed or submitted a form 80 with a prepayment amount enclosed.

Credit Card Number

Expiration Date – Expiration date on the credit card account.

INVOICE ATTRIBUTES:

Quantity Ordered – Quantity of the inventory item on the request.

Quantity Fulfilled – Quantity shipped.

BILL NOTIFICATION ATTRIBUTES:

Quantity Ordered – Quantity of the inventory item on the request.

Quantity Fulfilled – Quantity shipped.

Quantity Billed – Quantity of inventory item for which an amount is due from the customer.

Billed Amount – Amount in US dollars for the inventory item(s) listed on the invoice.

QUOTE ATTRIBUTES: (See form 72 doc)

ORDER ATTRIBUTES: (See form 72 doc)

BACK ORDER ATTRIBUTES (See form 72 doc)

10.3.3. Database Schema and Configuration

Database schema and other configuration data must include the following, but not limited to them. These reports can be generated from the CASE tools or prepared using the templates below.

Category	**Contents**
Schema	Database, Tablespace, Rollback Segment, Database Link
Table	Table, Check Constraints, Partition, Storage, Synonym, Snapshot, Cluster, Sequence
View	View
Column	Table/Column, Check constraints, Default value, Physical order
Index	Primary key, Foreign key, Alternate key, Partition, Storage
Referential Integrity(RI)	Primary key (Alter), Foreign key (On-delete), Unique key
Volume	Table Initial Volume, Expected Growth Rate
Stored Procedure	Stored procedures, Triggers
Roles	Roles, Profiles
Directory Structure	Installation Directory(folder) Structure and Contents
Initial Configuration	Initial Database Configuration Parameters

Oracle Database Schema Generated from ERWin
(Sample)

```
CREATE DATABASE "test2"
    CONTROLFILE REUSE
LOGFILE group 1 ('diskb:log1.log', 'diskc:log1.log') size 50K,
group 2 ('diskb:log2.log', 'diskc:log2.log') size 50K
    MAXLOGFILES 5
    MAXLOGHISTORY 100
DATAFILE 'diska:test2.dat'  size 2M
    MAXDATAFILES 10
    MAXINSTANCES 2
    ARCHIVELOG
    EXCLUSIVE
    ;

CREATE TABLESPACE "tabspace_2"
    DATAFILE 'diska:tabspace_2.dat'
    DEFAULT
    STORAGE (
    INITIAL 10K
    NEXT 50K
    MINEXTENTS 1
    MAXEXTENTS 999
    PCTINCREASE 10
        )
    ;
  CREATE TABLE "epark"."emp" (
"name"          VARCHAR2(20) NOT NULL,
  "address"       VARCHAR2(20) NULL
                  CONSTRAINT constraint3
                  CHECK (address=upper(address)),
CONSTRAINT "XPKemp"
    PRIMARY KEY ("name")
    )
    PCTFREE 20
    PCTUSED 80
TABLESPACE tabspace_2
    STORAGE (
    INITIAL 100GB
    NEXT 30GB
    MINEXTENTS 50
    MAXEXTENTS 100
    PCTINCREASE 10
    BUFFER_POOL KEEP
    )
    LOGGING
    CACHE
    ;

CREATE OR REPLACE VIEW "nhpview" ("name", "address", "symbol_code", "String", "xxx")  AS
SELECT "epark"."emp"."name", "epark"."emp"."address", "org"."symbol_code", "org_emp"."begin_date", end_date >'31-mar-2001'
    FROM "epark"."emp", "org", "org_emp"
    WHERE symbol_code='NHP';


create trigger tD_emp after DELETE on emp for each row
-- ERwin Builtin Thu Jul 19 14:47:35 2001
-- DELETE trigger on emp
declare numrows INTEGER;
    begin
/* ERwin Builtin Thu Jul 19 14:47:35 2001 */
/* emp belogns to org_emp ON PARENT DELETE RESTRICT */
    select count(*) into numrows
        from org_emp
        where
/*  %JoinFKPK(org_emp,:%Old," = "," and") */
    org_emp.name = :old.name;
```

```
if (numrows > 0)
then
raise_application_error(
-20001,
'Cannot DELETE emp because org_emp exists.'
);
end if;

-- ERwin Builtin Thu Jul 19 14:47:35 2001
end;
/
```

Database Schema/Configuration Templates

Database

Database	Logfile/Size	MaxLogFiles	MaxLogHistory	MaxLogMembers	Datafile/Size	MaxDataFiles	Controlfile reuse	Archivelog	MaxInstances
TEST2	"Group 1 ('diskb:log1.log','diskc:log1.log') size 50K" "Group 2 ('diskb:log2.log','diskc:log2.log') size 50K"	5	100		diska:test2.dat' size 2M	10	ReUse	Y	2

TableSpace

TableSpace	Datafile	Init	Next	PctIncrease	Minextents	Maxextents
TS2	‘diska:ts2.dat' size 2M	10K	50K	10	1	5
TS3	‘diska:ts3.dat' size 5M	50K	100K	10	1	10
TS4	‘diska:ts4.dat' size 10M	30K	50K	10	1	15

Rollback Segment

RollbackSeg	Public	TableSpace	Init	Next	PctIncrease	MinExtent	MaxExtent
RBS2	Yes	TB4	50K	50K	10	5	10
RBS_sam	No	TB4	5K	5K	10	5	10

Table Volume

Table	Owner	InitialRows	AvgRows	MaxRows	YrGrowthRate	RowSize	TableSpace	Init	Next	MinExtents	MaxExtents	PctIncrease	Cluster	Index-organized
holdings	nara	500	2000	9999	20%	124	TB2	6144	6144	1	5		No	No
titles	nara	60000	100000	999999	10%	24500	TB3	5M	5M	1	10		No	Yes
location	nara	500	1000	9999	20%	400	TB2	1M	5M	1	5		No	No

Table/Column

Table	Column	Data Type	Null?	Default	Domain	PK/FK	Constraints	Definition
employee	ssn	varchar2(9)	No			PK		employee's social security number
employee	grade	char(2)	No	5		FK		employee's grade
employee	salary	number(9)	Yes	0			sal_limit	employee's salary amount

Views

View	BaseTables	Description
empgr15_vw	employee	employee with grade=15
empsal-vw	employee	employee with salary >10000

Index

Index	Owner	Type	Unique	table	columns	Order	TableSpace	Init	Next	MinExtents	MaxExtent
empssn_I	nara2	PK	Yes	employee	ssn		ts_2	30M	5M	5	10
empgrsal_I	nara2		Yes	employee	grade	1	ts_2	30M	5M	5	10
				employee	salary	2					

Snapshot

Snapshot	Owner	TableSpace	Init	Next	PctIncrease	Refresh
emp_sf	nara3	tb2	50K	50K	50	Fast

Cluster

Cluster	Tables	Columns	Init	Next	PctIncrease
personnel	employee	deptno	100K	500K	10
	department	deptno			

Sequence

Sequence	Owner	Increment	Start	MaxValue	MinValue	Cycle	Cache	Order
empid_seq	nara	10	0	99999	0	Yes	10	Yes

Database Link

DBLink	Creator	Public	ConnectString	Description
mw_dbl	system	Yes	'd:mw'	Link to MW database
mw_sp_dbl	system	Yes	'd:mwsp'	Link to MW special media database

Synonym

Synonym	Public	Owner	Object	DBLink
emp	Yes	nara3	employee	hq

Roles

Role	Description	Profile	Users
sales_role	salesperson with update permission for the sales table	marketing_prof	Sam Joe

Profiles

Profile	Sessions/User	ConnectTime	Description
hr_prof	50	45	personnel user with resource limit

Role/Table/CRUD Matrix

holdings	R	C/R/U	R	C				
titles	R	R	R	C/R/U/D				
employee	C/R/U		R	C/R/U/D				
	sales_role	**Researcher_role**	**Public_role**	**Developer_role**				

Stored Procedure

Procedure	**Owner**	**Table**	**CRUD**	**Description**
Credit_sp	nara5	accounts	U	Update accounts with new balance

Installation Directory Structure

Disk/Directory	**Sub-directories**	**Sub-subdirectories**	**Files**	**File size**
C:\oracle1\ortest1\	admin	Bdump Udump Pfile create		
	control			
	data			
	redo			
d:\oracle2\ortest1\	Control			
	Data			
	redo			
d:\oracle3\ortest1\	Control			
	Data			
	Redo			
	exports			
e:\oracle4\ortest1\	data			
	arch			
e:\oracle5\ortest1\	data			

Key Database Initial Settings

Parameter Name	**Current Value**		
DB_BLOCK_SIZE	8192		
DB_BLOCK_BUFFERS	1000		
SHARED_POOL_SIZE	10000000		
SHARED_POOL_RESERVED_SIZE	1000000		
SORT_AREA_SIZE	1000000		
SORT_AREA_RETAINED_SIZE	1000000		
SORT_DIRECT_WRITES	TRUE		
LOCK_CHECKPOINT_INTERVAL	9999999		
OPEN_CURSORS			
DBWR_IO_SLAVES			
LOG_BUFFER	1000000		
BUFFER_POOL_KEEP			
BUFFER_POOL_RECYCLE			

DB_FILE_MULTIBLOCK_READ_COUNT			
NLS_DATA_FORMAT	“DD-MON-YYYY”		
OPTIMIZER_MODE	CHOOSE		
LOG_ARCHIVE_DUPLEX_DEST			
LOG_ARCHIVE_MIN_SUCCEED_DEST	1		
LOCK_SGA	TRUE		
PRE_PAGE_SGA	TRUE		
LOCK_SGA_AREAS	FALSE		
LARGE_POOL_SIZE	1000000		
LARGE_POOL_MIN_ALLOC			
LOG_SIMULTANEOUS_COPIES			